

SIMULATION-BASED STOCHASTIC OPTIMIZATION ON DISCRETE DOMAINS:  
INTEGRATING OPTIMAL COMPUTING AND RESPONSE SURFACES


by

Mark W. Brantley  
A Dissertation  
Submitted to the  
Graduate Faculty  
of  
George Mason University  
In Partial Fulfillment of  
The Requirements for the Degree  
of  
Doctor of Philosophy  
Information Technology

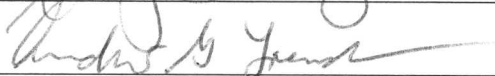
Committee:




Dr. Chun-Hung Chen, Dissertation Director



Dr. Kathryn Laskey, Committee Member



Dr. Andrew Loerch, Committee Member



Dr. Brian Mark, Committee Member



Dr. Daniel Menascé, Senior Associate Dean



Dr. Lloyd J. Griffiths, Dean, Volgenau  
School of Engineering

Date: 3/31/2011

Spring Semester 2011  
George Mason University  
Fairfax, VA

Simulation-based Stochastic Optimization on Discrete Domains: Integrating Optimal  
Computing and Response Surfaces

A dissertation submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy at George Mason University

By

Mark W. Brantley  
Master of Science  
Rensselaer Polytechnic Institute, 1998  
Bachelor of Science  
United States Military Academy, 1988

Director: Chun-Hung Chen, Professor  
Department of Systems Engineering and Operations Research

Spring Semester 2011  
George Mason University  
Fairfax, VA

Copyright: 2011 Mark W. Brantley  
All Rights Reserved

## DEDICATION

This is dedicated to my grandmother, Therese Anne Cedotal Abadie. She would have been proud.

## ACKNOWLEDGEMENTS

I would like to thank the members of the committee for their time and guidance. In particular, I would like to thank Professor Chen for his patience, understanding, and insight. I would also like to acknowledge the assistance that Dr. RK Jana provided during the formulation of the research effort as well as the assistance that Professor Loo Hay Lee (National University of Singapore), Professor Argon Chen (National Taiwan University), and Professor Douglas J. Morrice (The University of Texas at Austin) provided as co-authors of conference and journal papers. I have also benefitted from the feedback provided by area editor Professor Shane Henderson and the anonymous referees at *Operations Research* as well as by department editor Professor Enver Yucesan and the anonymous referees at *IIE Transactions*. I would also like to thank my fellow student LTC John Ferguson for his friendship and assistance along this path. Lastly, I would like to thank my wife, daughter, and parents for their patience, love, and encouragement.

Significant portions of this dissertation are derived from two journal papers, Brantley et al. (2010) and Brantley et al. (2011), and use of this copyrighted material has been permitted.

This dissertation also contains portions of material from the following papers with copyright permissions: Brantley and Chen (2005), Brantley, Lee, Chen, and Chen (2008), Morrice, Brantley, and Chen (2008), Morrice, Brantley, and Chen (2009).

## TABLE OF CONTENTS

	Page
List of Tables .....	vi
List of Figures .....	vii
List of Abbreviations/Symbols .....	viii
Abstract .....	xi
Chapter 1 Introduction .....	1
Chapter 2 Problem Setting .....	10
2.1 Problem Statement .....	10
2.2 A Bayesian Regression Framework .....	15
2.3 Reducing the Number of Comparisons .....	19
Chapter 3 Optimal Simulation Design Method .....	25
3.1 A Lower Bound for PCS .....	25
3.2 Sufficient Number of Support Points .....	26
3.3 Optimal Allocations .....	31
3.4 Approximately Optimal Allocations .....	36
3.5 Optimal Support Point Location .....	38
3.6 OSD Procedure .....	39
3.7 Numerical Testing and Results .....	41
Chapter 4 OSD for Partitioned Domains .....	55
4.1 Problem Statement and Framework .....	56
4.2 A Lower Bound for PCS .....	61
4.3 Lagrangian Formulation for APCS .....	67
4.4 Approximately Optimal Allocations .....	70
4.5 POSD Procedure .....	77
4.6 Numerical Testing and Results .....	79
Chapter 5 Conclusions and Future Work .....	89
Appendices .....	94
Appendix A Proof of Theorem 4 .....	95
Appendix B Proof of Theorem 5 .....	100
Appendix C ProModel Code for OSD (one partition case) .....	106
Appendix D Proofs of Lemma 4 and Lemma 5 .....	130
Appendix E Proofs of Lemma 6 and Lemma 7 .....	134
Appendix F Proof of Proposition 1 .....	138
Appendix G ProModel Code for Partitioning OSD (POSD) .....	142
List of References .....	174

LIST OF TABLES

Table	Page
Table 1: OSD Allocations for Experiment 1.....	46
Table 2: OSD Allocations for Experiment 2.....	48
Table 3: Results of Experiment 6.....	54
Table 4: Results of Experiment 8.....	84

## LIST OF FIGURES

Figure	Page
Figure 1: General Simulation Study Steps.....	3
Figure 2: Two Comparisons (Interior Case) .....	23
Figure 3: Two Comparisons (Boundary Case) .....	24
Figure 4: Location of $x_s$ for the first case in Theorem 5 .....	39
Figure 5: Results of Experiment 1 .....	45
Figure 6: Results of Experiment 2 .....	47
Figure 7: Results of Experiment 3 .....	49
Figure 8: Results of Experiment 4 .....	50
Figure 9: Results of Experiment 5 .....	52
Figure 10: Intuitive Lower Bound Scenarios.....	65
Figure 11: $y(x_i) = \sin(x_i) + \sin(10x_i/3) + \ln(x_i) - 0.84x_i + 3$ .....	82
Figure 12: Results of Experiment 7 .....	83
Figure 13: Results of Experiment 9 .....	86
Figure 14: $y(x_i) = 10x_i + 10/x_i$ .....	87
Figure 15: Results of Experiment 10 .....	88



## LIST OF ABBREVIATIONS/SYMBOLS

APCS	Approximate Probability of Correct Selection
DOE	Design of Experiments
D-opt	D-optimality Criterion
EA	Equal Allocation
EA-RS	Equal Allocation – Response Surface
MARS	Multivariate Adaptive Regression Splines
OCBA	Optimal Computing Budget Allocation
OSD	Optimal Simulation Design
PCS	Probability of Correct Selection
POSD	Partitioning Optimal Simulation Design
RSM	Response Surface Methodology
RV	Random Variable
R&S	Ranking and Selection

For brevity, we present the notation for the one partition case. If not indicated otherwise, the notation for the partitioned case follows by adding a subscript  $h$  to indicate the partition number.

$a$	Variable to indicate the stationary point of the unknown underlying equation
$A$	Index for one of two comparisons (left comparison)
$b$	Index to indicate the best design location
$B$	Index to indicate the partition with the global best design location
$c$	Vector of constants
$C$	DOE criterion
$d(x_i)$	Difference between $y(x_i)$ and $y(x_b)$ (used for within partition comparisons)
$\hat{d}(x_i)$	Difference between $\hat{y}(x_i)$ and $\hat{y}(x_b)$
$\tilde{d}(x_i)$	RV whose probability distribution is the posterior distribution of $d(x_i)$
$D$	Lagrange interpolating polynomial coefficient for within partition comparisons
$E$	Lagrange interpolating polynomial coefficient for between partition comparisons
$E[\bullet]$	Expected value
$f(x_i)$	Simulation output with noise
$F$	$n$ dimensional vector containing the replication output measures $f(x_i)$
$g(x_i)$	Linear function in Theorem 1 defined as $g(x_i) = [\tilde{\beta}_1 + \tilde{\beta}_2(x_i + x_b)]$

$G$	$(n + l)$ dimensional vector containing the replication output measures $f(x_i)$
$h$	Partition number
$I$	Identity matrix
$j$	Index variable
$k$	Number of designs in a partition
$l$	Number of additional simulation runs
$L$	Lower bound for the APCS
$m$	Number of partitions
$M$	Index variable for the most competitive comparison
$n$	Number of simulation runs already conducted
$N_i$	Number of simulation runs allocated to $x_i$ (one partition problem)
$N_{hi}$	Number of simulation runs for partition $h$ allocated to $x_{hi}$
$N_{h\bullet}$	Number of simulation runs allocated to partition $h$
$p(\bullet)$	Probability (lower case used in Bayesian regression derivation)
$P\{\bullet\}$	Probability (upper case used in OSD/POSD derivations)
$P_{hM}$	Probability of the most competitive comparison for partition $h$
$P_\Omega$	Probability of the most competitive between partition comparison
$q$	Coefficients for the optimal allocations
$Q$	Lagrangian function
$r$	Used in a change of variables for integration
$R_h$	Signal to noise ratio of the most competitive comparison for partition $h$
$T$	Total number of simulation runs in the simulation budget
$t$	Superscript for the transpose of a matrix
$U$	Upper bound for the APCS
$v$	Used in integration
$w$	Variable used for sign function
$W$	$(n + l) \times 3$ matrix composed of rows consisting of $[1 \ x_i \ x_i^2]$
$x_i$	Design location $i$
$x_1$	Design location at the left boundary of the domain
$x_k$	Design location at the right boundary of the domain
$x_s$	Design location selected as the middle support point
$x_b$	Design location with the smallest least squares estimate
$x_{Bb}$	Design location with the smallest least squares estimate (globally)
$x_{hb}$	Design location with the smallest least squares estimate for partition $h$
$X$	$n \times 3$ matrix composed of rows consisting of $[1 \ x_i \ x_i^2]$
$X_i$	Vector consisting of $[1 \ x_i \ x_i^2]$
$y(x_i)$	Unknown underlying function of simulation output, $E[f(x_i)]$
$\hat{y}(x_i)$	Least squares estimate for $y(x_i)$

$\tilde{y}(x_i)$	RV whose probability distribution is the posterior distribution of $y(x_i)$
$Z$	Index for one of two comparisons (right comparison)
$\alpha_i$	Percentage of the simulation budget allocated to $x_i$ (one partition problem)
$\alpha_{hi}$	Percentage of the budget for partition $h$ allocated to $x_{hi}$
$\beta$	Vector of unknown parameters of the quadratic underlying equation
$\hat{\beta}$	Vector of least squares estimates for $\beta$
$\beta_0$	Constant term in the underlying equation
$\hat{\beta}_0$	Least squares estimate for the constant term $\beta_0$
$\beta_1$	Linear term in the underlying equation
$\hat{\beta}_1$	Least squares estimate for the linear term $\beta_1$
$\beta_2$	Quadratic term in the underlying equation
$\hat{\beta}_2$	Least squares estimate for the quadratic term $\beta_2$
$\tilde{\beta}$	RV whose probability distribution is the posterior distribution of $\beta$
$\gamma$	Index for a particular support point and allocation scheme,
$\Gamma$	Variable for the steepest descent sum
$\hat{\delta}(x_{hb})$	Least squares estimate for $\tilde{\delta}(x_{hb})$
$\tilde{\delta}(x_{hb})$	Difference between $\tilde{y}(x_{hb})$ and $\tilde{y}(x_{Bb})$ (between partition comparisons)
$\Delta$	Spacing between the design locations (assumed constant)
$\varepsilon$	Vector of simulation replication noise terms
$\varepsilon_j$	Noise of the underlying equation for simulation replication $j$
$\theta_j$	The number runs allocated during iteration $j$ in the implementing algorithm
$\lambda$	Lagrangian multiplier
$\mu$	Mean of the exponential distribution
$\xi_{hb}$	Variance of $\tilde{\delta}(x_{hb})$
$\rho$	Probability of success of one trial in the binomial distribution
$\sigma^2$	The variance of the simulation output
$\varsigma_i$	Variance of $\tilde{d}(x_i)$
$\tau_1$	Lower limit of the uniform distribution
$\tau_2$	Upper limit of the uniform distribution
$\phi$	Normal distribution cumulative distribution function
$\psi$	Set of partitions with the lower bounds established using quadratic information
$\Psi$	Variable to indicate the number of simulation runs in Lemma 2
$\omega$	Sum of the squares of the error terms for the least squares estimate

## ABSTRACT

### SIMULATION-BASED STOCHASTIC OPTIMIZATION ON DISCRETE DOMAINS: INTEGRATING OPTIMAL COMPUTING AND RESPONSE SURFACES

Mark W. Brantley, PhD

George Mason University, 2011

Dissertation Director: Dr. Chun-Hung Chen, Professor

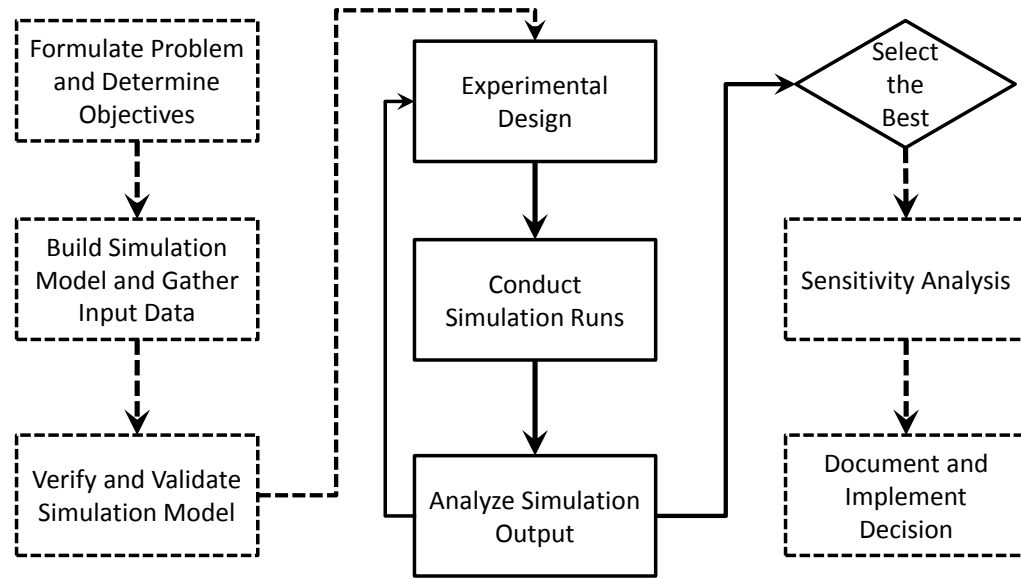
Simulation can be a very powerful tool to help decision making in many applications but exploring multiple courses of actions can be time consuming. Numerous ranking & selection (R&S) procedures have been developed to enhance the simulation efficiency of finding the best design. This dissertation explores the potential of further enhancing R&S efficiency by incorporating simulation information from across the domain into a regression metamodel. Under some common conditions in most regression-based approaches, our new method provides approximately optimal rules that determine the design locations to conduct simulation runs and the number of samples allocated to each design location for problems with only one partition. In addition to utilizing concepts from the design of experiments (DOE) literature, it introduces the probability of correct selection (PCS) optimality criterion that underpins our new R&S method to the DOE literature. This dissertation then extends the method by incorporating simulation information from across a partitioned domain into a regression based metamodel. Our

new method provides approximately optimal rules for between and within partitions that determine the number of samples allocated to each design location. Numerical experiments demonstrate that our new approaches for one partition domains and for multiple partition domains can dramatically enhance efficiency over existing efficient R&S methods.

## CHAPTER 1 INTRODUCTION

Simulation is a popular tool for designing large, complex, stochastic engineering systems, since closed form analytical solutions generally do not exist for such problems. Simulation allows one to accurately specify a system through the use of logically complex, and often non-algebraic, variables and constraints. Detailed dynamics of complex, stochastic systems can therefore be modeled. This capability complements the inherent limitation of traditional optimization. Semiconductor manufacturing system is a good example which is characterized by complex and reentrant production processes over many heterogeneous machine groups with stringent performance requirements. Semiconductor manufacturing faces stringent challenges of volatile product demands, very short time to market, complex but fast evolving process technology, sky-rocketing capital investment and highly cost-sensitive competition (Hsieh et al. 2001, 2007). The health care arena provides another example. Simulations provide the ability to analyze the complex decisions associated with patient flow and ambulance planning (Henderson and Mason 2005 and Zeto et al. 2005). As a final example, simulations can help determine how to procure military equipment. These simulations can provide a means to investigate the trade-offs of performance, cost, and reliability given environmental and operating scenarios (Brantley et al 2002).

Although simulation can help design good systems for efficient operations for such complex systems as the examples provided above, the added flexibility of simulation often creates models that are computationally intractable. Simulation optimization is a method to find a design consisting of a combination of input decision variable values of a simulated system that optimizes a particular output performance measure or multiple performance measures of the system. For instance, using the ambulance planning example from above, a decision maker may want to determine the optimal locations to station a fixed number of ambulances in order to minimize the average emergency response time. Figure 1 below provides general steps involved in conducting a simulation study (based upon Harrell et al (1995), Law and Kelton (2000), and Banks et al. (2001) for steps in constructing simulation models in particular and Giordano and Weir (1985) and Clemen (1996) for steps in constructing models in general). Note that while this figure is presented in a sequential manner with only one feedback loop, a simulation study may have feedback loops between all stages shown in the figure (Law and Kelton 2000). Simulation optimization is associated with the steps from determining the experimental design to selecting the best design.



**Figure 1: General Simulation Study Steps**

The level of complexity associated with a simulation optimization problem is dependent upon the nature of the input decision variables, the nature of the underlying function associated with the output performance measure of the system, and the resources available to solve the problem. For example, the input decision variables may be discrete or continuous; and one dimensional or multi-dimensional. The underlying function associated with the output decision variables may be deterministic or stochastic; discrete or continuous; and linear or nonlinear. The resource considerations may include the time available or the size of the computing budget dedicated to solving the problem as well as the number of computer platforms available. This dissertation investigates stochastic problems on a discrete domain with a finite simulation budget consisting of runs conducted sequentially on a single computer. To assess the performance at a single design location on the domain, the uncertainty in the system performance measure



requires multiple runs to obtain a good estimate of the performance measure. Thus, for a simulation that may require hours or days to conduct a single iteration, the simulation time required to conduct multiple runs for a large number of design locations may be cost prohibitive.

The problem we consider is that of selecting the best design from among the finite number of choices. Ranking and Selection (R&S) procedures are statistical methods specifically developed to select the best design or a subset that contains the best design from a relatively small set of  $k$  competing design alternatives (Goldsman and Nelson 1994). Dudewicz and Dalal (1975) and Rinott (1978) developed two-stage procedures for selecting the best design or a design that is very close to the best system. Many researchers have extended this idea to more general ranking-and-selection settings in conjunction with new developments (e.g., Bechhofer, Santner, and Goldsman 1995).

To improve efficiency for R&S, several approaches have been explored for problems of selecting a single best design. Intuitively, to ensure a high PCS, a larger portion of the computing budget should be allocated to those designs that are critical in the process of identifying the best design. A key consequence is the use of both the means and the variances in the allocation procedures, rather than just the variances, as in Dudewicz and Dalal (1975) and Rinott (1978). Among examples of such approaches, the Optimal Computing Budget Allocation (OCBA) approach by Chen et al. (2000, 2008) is the most relevant to this dissertation. OCBA maximizes a simple heuristic approximation of the PCS. The approach by Chick and Inoue (2001ab) estimates the PCS with Bayesian posterior distributions and allocates further samples using decision-theory tools to

maximize the expected value of information in those samples. In a similar myopic effort, Frazier et al. (2008) develop the knowledge gradient approach that uses independent multivariate normal priors to solve Bayesian R&S problems and then extend the method for problems that assume a correlated variance structure between each design (Frazier et al., 2009). The procedure by Kim and Nelson (2006) allocates samples in order to provide a guaranteed lower bound for the frequentist PCS integrated with ideas of early screening. Goldsman et al. (2005) and Fu et al. (2005) discuss the general concepts of several more approaches while Branke et al. (2005, 2007) provide a nice overview and an extensive comparison for some of the aforementioned selection procedures.

This dissertation takes a new approach to further improve the efficiency for R&S by incorporating information from neighboring designs or information from estimates of the underlying function generating the data. In that sense, it is related to the Bayesian global optimization methods that sequentially search for design points in order to efficiently find the optimal values of continuous functions. Some of these methods take individual or a small number of samples at locations in order to build a metamodel across the domain. For example, see Huang et al. (2006) and Villemonteix et al. (2009) for using Kriging interpolation to solve general global optimization problems and van Beers and Kleijnen (2003) for applying Kriging interpolation to a stochastic simulation setting. Other methods assume varying degrees of structure to the problem such as correlated variance terms. Similar to the previously mentioned R&S method by Frazier et al. (2009), Calvin and Žilinskas (2005) extend the work of Kushner (1964) in order to use Wiener process priors to address stochastic global optimization problems. Our approach is

developed for problems where we can assume an approximately quadratic form of the underlying structure or portions of the underlying structure so that we can use DOE to capture that information.

DOE is a commonly used approach for gathering information when variation is present, and can be categorized into three branches (Melas 2006). The first DOE branch is based upon combinatorial principles such as Latin squares. Sanchez (2005) provides an overview of this method for simulation experiments and provides a list of types of experimental designs. Chen and Cheng (2006ab) use this approach to minimize the variance of estimated local stationary points in order to construct a ridge path across the domain. The second DOE branch is the response surface methodology (RSM). This method constructs a metamodel using a regression equation and finds the optimal value using a local search method, typically based upon the gradient of the estimated function (cf. Neddermeijer et al. 2000). We will primarily use concepts from the third DOE branch called optimal experimental design that determines design locations and allocates experimental samples according to specific optimality criteria. Federov and Muller (1997), Cheng, Melas, and Pepelyshev (2001), and Melas, Pepelyshev, and Cheng (2003) develop optimal designs for estimating the extreme point in quadratic regression. From a simulation perspective, Barton (2005) in his discussion of forward-inverse metamodels suggests that optimal experimental designs offer great opportunities for use in simulation optimization but that past research has only focused upon leveraging the D-optimality criterion. Cheng and Kleijnen (1999) introduce the concept of applying the concepts of DOE to simulation optimization and develop a criterion that provides the best fitting

polynomial for queuing models. Lamb and Cheng (2002) extend this method for a generalized regression metamodel and derive an optimal allocation for the goodness of fit criterion.

The method proposed here, called optimal simulation design (OSD), takes an approach that is different than most R&S methods by incorporating information from across the domain into a regression equation. Unlike traditional R&S methods, this regression based approach requires simulation of only a subset of the alternative design locations and so the simulation efficiency can be dramatically enhanced. To efficiently utilize the simulation budget, we want to determine i) which designs should be selected for simulation; ii) the number of simulation runs for those selected designs. The goal is to maximize the probability of correctly selecting the best design (PCS). To our best knowledge, none of the DOE literature has addressed the first question directly for the PCS criterion. While the second part is similar to some R&S methods such as OCBA, the problem is much more complex because of the use of a regression metamodel. This dissertation develops an OSD method to address both questions. Numerical testing shows that the use of regression metamodel can indeed dramatically enhance simulation efficiency, even compared with some existing efficient R&S methods such as OCBA. As compared with the use of regression metamodels, the OSD methods offer a further improvement over not only naïve response surface methods (by 50~70% reduction) but also the well known D-optimality approach in DOE literature (by another 22%~28% reduction).

While the use of a regression metamodel can dramatically enhance efficiency, the OSD also inherits some typical assumptions from most DOE approaches. It is assumed that there is an underlying quadratic function for the means and the simulation noise is homogeneous across the domain of interest. Such assumptions are common in some of the DOE literature as well as when applying an iterative search method (e.g., Newton's method in nonlinear programming) and focusing on a small local area of the search space in each iteration. After solving our problem for just one partition across the entire domain, this dissertation then expands the use of OSD by determining approximately optimal allocations and support points for a domain that has been partitioned such that the underlying function for each partition is approximately quadratic.

While enhancing R&S efficiency by incorporating simulation information from across the domain into a regression metamodel has been explored in previous literature, the contributions of this dissertation are fourfold. First, we develop approximately optimal rules that determine the design locations to conduct simulation runs and the number of samples allocated to each design location for problems on a single partition. Numerical examples reinforce the derived results. Secondly, we advance the DOE literature by considering a new criterion -- PCS -- and offering an optimal sampling strategy for this new criterion. This dissertation then extends the method by incorporating simulation information from across a partitioned domain into a regression based metamodel. In doing so, we thirdly develop a framework to integrate OSD with partitioning methods for general simulation optimization problems where we can assume the OSD assumptions hold for each partition. Given design locations on a partitioned

domain, we want to utilize the simulation budget in a most efficient way by determining the number of simulation runs allocated between partitions and also for the design locations within each partition. Finally, we provide a heuristic approximation of the PCS that performs well in numerical testing. Our new method provides approximately optimal rules for between and within partitions that determine the number of samples allocated to each design location.

The rest of the dissertation is organized as follows. Chapter 2 introduces the simulation optimization problem setting, the Bayesian regression framework, and then shows we can reduce the number of comparisons in our problem based upon our problem assumptions. Chapter 3 provides the development of the OSD method and provides the results of numerical experiments comparing the new OSD method and other methods. Chapter 4 extends the OSD method for applications on partitioned domains and then also provides results from numerical experiments using the new partitioning OSD (POSD) method and other methods. Finally, Chapter 5 provides the conclusions and suggestions for future work using the concepts introduced here.

## CHAPTER 2 PROBLEM SETTING

### 2.1. Problem Statement

This dissertation explores a problem with the principal goal of selecting the best of  $k$  alternative design locations in a one-dimensional space. Without loss of generality, we consider the minimization problem shown below where the “best” design location is the one with the smallest expected performance measure.

$$\min_{x_i} y(x_i) = E[f(x_i)]; x_i \in [x_1, x_2, \dots, x_k]. \quad (1)$$

It is important to note that many simulation optimization papers refer to the alternatives or configurations under consideration as “designs”. We adopt a slightly different convention. We consider the case of simulation output that is produced by an unknown function of one variable at  $k$  design locations where  $x_i < x_{i+1}$ ,  $i = 1, \dots, k$ . Therefore, we will refer to the alternatives under consideration as “design locations” to reflect that there is a cardinal relationship between the alternatives.

In this chapter, we consider that the expectation of the unknown underlying function is quadratic or approximately quadratic in nature on the prescribed domain, i.e.,

$$y(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2. \quad (2)$$

However, the parameters  $\beta$  are unknown. We consider a common case where  $y(x_i)$  must be estimated via simulation with noise and that the simulation output  $f(x_i)$  is independent from replication to replication such that, for replication  $j$ ,

$$f(x_i) = y(x_i) + \varepsilon_j; \quad i = 1, \dots, k, \text{ where } \varepsilon_j \sim N(0, \sigma^2). \quad (3)$$

The parameters  $\beta$  are unknown so  $y(x_i)$  are also unknown. However, we can find an estimated expected performance measure at  $x_i$ , that we define as  $\hat{y}(x_i)$ , by using a least squares estimate of the form shown in (4) below where  $\hat{\beta}_0$ ,  $\hat{\beta}_1$ , and  $\hat{\beta}_2$  are the least squares parameter estimates for the corresponding parameters associated with the constant, linear, and quadratic terms in (2).

$$\hat{y}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2 \quad (4)$$

For ease of notation, we will define  $\hat{\beta} = [\hat{\beta}_0 \quad \hat{\beta}_1 \quad \hat{\beta}_2]$ .

In order to obtain the least squares parameter estimates, we take samples on any choice of  $x_i$  (on at least three design locations to avoid a singular solution). We assume that these  $x_i$  are given beforehand and we can only take samples from these points. We use a matrix notation for linear regression consistent with those used Draper and Smith (1998) and Neter, et al. (1996). Given a total of  $n$  samples, we define  $F$  as the  $n$  dimensional vector containing the replication output measures  $f(x_i)$  and  $X$  as the  $n \times 3$  matrix composed of rows consisting of  $[1 \quad x_i \quad x_i^2]$  with each row corresponding to its respective entry of  $f(x_i)$  in  $F$ . Using the matrix notation and a superscript  $t$  to indicate the transpose of a matrix and then following an approach commonly shown in regression



texts, we determine the least squares estimate for the parameters  $\beta$  which minimize the sum of the squares of the error terms  $\omega = (F - X\beta)'(F - X\beta)$ .

Simplifying, we obtain the equation below.

$$\omega = F'F - 2\beta X'F - \beta X'X\beta$$

Differentiating we obtain

$$\frac{\partial \omega}{\partial \beta} = X'F - X'X\beta.$$

This leads to the normal equation of:

$$X'X\beta = X'F.$$

In the normal equation, the matrix  $X'X$  is called the information matrix for  $\beta$ . Since we are using a quadratic model,

$$X'X = \begin{bmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}$$

Solving the normal equations, we obtain the least squares estimate for the parameters as shown below:

$$\hat{\beta} = (X'X)^{-1} X'F.$$

Since the assumptions associated with (2) and (3) satisfy the Gauss-Markov conditions, these parameter estimates that minimize the sum of the squared residuals are unbiased and have the minimum variance for all unbiased linear estimators (Draper and Smith, 1998).

Given this model defined by (2) and (3), the type of optimality criteria used will determine the way that we allocate the simulation runs. There are numerous optimality criteria that can be imposed – typically identified by a particular letter or combination of letters. The most basic criterion is that of D-optimality which minimizes the variance of the regression parameters. Mathematically, this equates to an allocation scheme being D-optimal if it maximizes the value of  $|X'X|$ . Other examples include the G-optimality criterion that minimizes the maximum of the standardized variances over the domain, the c-optimality criterion that minimizes a linear combination of the parameters  $c'\beta$ , and DS-optimality that minimizes the variance of a subset of the regression parameters (Atkinson and Donev, 1998). This framework is similar to the criteria used by the various R&S methods described in Chapter 1.

Our problem is different from all existing DOE optimality criteria and R&S methods. We aim to select the design location associated with the smallest mean performance measure from among the  $k$  design locations within the constraint of a total computing budget with only  $T$  simulation replications. Given the least squares estimates for the parameters, we can use (4) to estimate the expected performance measure at each design location. We designate the design location with the smallest least squares estimate as  $x_b$  so that  $\hat{y}(x_b) = \min_i \hat{y}(x_i)$ . Given the uncertainty of the estimate of the underlying function,  $x_b$  is a random variable that is dependent upon the size of the computing budget and the allocations to each design location. We define Correct Selection as the event where  $x_b$  is indeed the best location and we define  $N_i$  as the number of simulation

replications conducted at design location  $x_i$ . Since the simulation is expensive and the computing budget is restricted, we seek to develop an allocation rule for each  $N_i$  in order to provide as much information as possible for the identification of the best design location. Our goal then is to determine the optimal allocations to the design locations that maximize the probability that we correctly select the best design (*PCS*). This OSD problem is reflected in (5) below.

$$\begin{aligned} \max_{N_1, \dots, N_k} \quad & PCS = P\{y(x_b) \leq y(x_i) \forall i\} \\ \text{s.t.} \quad & N_1 + N_2 + \dots + N_k = T \end{aligned} \tag{5}$$

The constraint  $N_1 + N_2 + \dots + N_k = T$  denotes the total computational cost and implicitly assumes that the simulation execution times per sample are constant across the domain.

The nature of this problem makes it extremely difficult to solve. As per (3), to understand the underlying function  $y(x_i)$ , we must conduct simulation runs to estimate  $f(x_i)$ , which is a measure of the system performance. This is compounded by the fact that  $f(x_i)$  is a function of the random variable  $\varepsilon$ . To even assess the performance at one point on the domain, the uncertainty in the system performance measure requires multiple runs to obtain good approximations of the performance measure. Since the optimal allocation is dependent upon the uncertainty of the parameters and the random variable  $x_b$ , we can only estimate the *PCS* even after exhausting the total simulation budget  $T$ . Incorporating the information from the underlying function adds an additional level of complexity to the derivation of the optimal allocations; however, it is this concept that we

aim to exploit in order to provide a significant improvement in the ability to maximize *PCS*.

## 2.2. A Bayesian Regression Framework

In order to solve the problem in (5), we must obtain estimates for the parameters  $\beta$ . Assuming that the conditional distribution of the simulation output vector  $F$  is a multivariate normal distribution with mean  $X\beta$  and a covariance matrix  $\sigma^2 I$  where  $I$  is an identity matrix, we can express the conditional probability of the simulation run output as shown in (6) below.

$$p(F | \beta, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left[\frac{-1}{2\sigma^2} (F - X\beta)' (F - X\beta)\right]. \quad (6)$$

For ease of derivation, we assume that  $\sigma^2$  is known. Degroot (1970), Law and Kelton (2000), and Chen and Lee (2010) discuss other approaches for when  $\sigma^2$  is not known. Without loss of generality, in Chapter 3, we find an approximate solution to our budget allocation problem that does not depend upon the value of  $\sigma^2$  and numerical experiments for our method developed in Chapter 4 demonstrate that the method performs well even when using an estimate of the variance.

Due to the ease of the derivation, we will proceed with a Bayesian regression framework where the parameters  $\beta$  are assumed to be unknown and are treated as random variables. We aim to find the posterior distribution of  $\beta$  as the simulation replications are conducted and use this distribution to update the posterior distribution of

the performance measures for each design location. We can then perform the comparisons with the performance measure at design location  $x_b$  as expressed in (5).

We begin by noting that the conditional probability expressed in (6) can be decomposed as shown in (7) below.

$$p(F | \beta, \sigma^2) = \frac{p(\beta, F, \sigma^2)}{p(\beta, \sigma^2)} = \frac{1}{p(\beta | \sigma^2)} \frac{p(\beta, F, \sigma^2)}{p(\sigma^2)}. \quad (7)$$

In a similar manner, given a set of  $n$  initial simulation runs with the output contained in vector  $F$ , the posterior distribution of  $\beta$  can be expressed as below.

$$p(\beta | F, \sigma^2) = \frac{p(\beta, F, \sigma^2)}{p(F, \sigma^2)} = \frac{1}{p(F | \sigma^2)} \frac{p(\beta, F, \sigma^2)}{p(\sigma^2)}. \quad (8)$$

Rearranging the terms from (7) and (8), we obtain Bayes' law that expresses the posterior distribution of  $\beta$  as shown below.

$$p(\beta | F, \sigma^2) = \frac{p(F | \beta, \sigma^2)p(\beta | \sigma^2)}{p(F | \sigma^2)}.$$

As noted by Gill (2002), the  $p(F | \sigma^2)$  term in the denominator does not provide any relevant information about the parameters  $\beta$ . This term acts a normalizing constant to ensure the conditional probability sums to one. For convenience, we will omit this term and work with a kernel as expressed in (9) below.

$$p(\beta | F, \sigma^2) \propto p(F | \beta, \sigma^2)p(\beta | \sigma^2). \quad (9)$$

We assume the case presented by DeGroot (1970) where we have an improper prior distribution with little prior knowledge about  $\beta$  such that

$$p(\beta | \sigma^2) \propto \frac{1}{\sigma^2}. \quad (10)$$

Substituting the results of (6) and (10) into (9), we obtain

$$p(\beta | F, \sigma^2) \propto \sigma^{-n-2} \exp \left[ \frac{-1}{2\sigma^2} \{ (F - X\beta)' (F - X\beta) \} \right]. \quad (11)$$

Therefore, the mean and variance of the parameters  $\beta$  are

$$\begin{aligned} E(\beta | F, \sigma^2) &= (X'X)^{-1} X'F \\ \text{cov}(\beta | F, \sigma^2) &= \sigma^2 (X'X)^{-1}. \end{aligned} \quad (12)$$

As noted by DeGroot (1970) and shown by Gill (2002), this same result can be obtained by assuming a conjugate normal prior distribution and applying an asymptotic assumption such that the number of simulation runs becomes very large.

For ease of notation, we will use  $\tilde{\beta}$  and  $\tilde{y}(x_i)$  to denote the random variables whose probability distributions are the posterior distribution of  $\beta$  and  $y(x_i)$  conditional on  $F$  given samples respectively. We can then use the posterior distribution of  $\beta$  to update the posterior distribution of the performance measures for each design location such that

$$\tilde{y}(x_i) = \tilde{\beta}_0 + \tilde{\beta}_1 x_i + \tilde{\beta}_2 x_i^2.$$

Therefore, given a set of  $n$  initial simulation runs conducted at the design locations described in  $X$  with the output contained in vector  $F$  and the design location  $x_b$  obtained from the least squares results derived in the previous section, we can

redefine the probability of correct selection from (5) based on the Bayesian concept (Chen et al. 2010 and He et al. 2007) as

$$PCS = P\{\tilde{y}(x_b) \leq \tilde{y}(x_i) \forall i\}. \quad (13)$$

The posterior distribution of  $\beta$  is given by

$$\tilde{\beta} \sim N[(X^T X)^{-1} X^T F, \sigma^2 (X^T X)^{-1}]. \quad (14)$$

We define  $X_i$  as the vector consisting of  $[1 \ x_i \ x_i^2]$ . Since  $\tilde{y}(x_i)$  is a linear combination of the  $\tilde{\beta}$  elements, this means that  $\tilde{y}(x_i)$  has a Gaussian distribution of the form

$$\tilde{y}(x_i) | X, F \sim N[X_i (X^T X)^{-1} X^T F, \sigma^2 X_i^T (X^T X)^{-1} X_i]. \quad (15)$$

We are interested in how the PCS in (13) changes if we conduct  $l$  additional simulation runs before we actually conduct the simulation replications so that we can make allocations that maximize the PCS in (5). Given the  $l$  additional simulation runs, we define  $G$  as the  $(n+l)$  dimensional vector containing the replication output measures  $f(x_i)$  from the original set of simulations runs and the  $l$  additional simulation runs.

Likewise, we define  $W$  as the  $(n+l) \times 3$  matrix composed of rows consisting of

$[1 \ x_i \ x_i^2]$  with each row corresponding to its respective entry of  $f(x_i)$  in  $G$ . The

updated posterior distribution of the performance measures for each design location

becomes analogous to (15):

$$\tilde{y}(x_i) | W, G \sim N[X_i (W^T W)^{-1} W^T G, \sigma^2 X_i^T (W^T W)^{-1} X_i].$$

When  $l \ll n$ , we assume that  $(X'X)^{-1}X'F \sim (W'W)^{-1}W'G$  such that our predictive posterior distribution is approximately

$$\tilde{y}(x_i) \sim N[X_i(X'X)^{-1}X'F, \sigma^2 X_i'(W'W)^{-1}X_i]. \quad (16)$$

The change in the variance after these additional runs is one of the key elements of our derivation and will be explained in greater detail in Chapter 3. Suffice it to say that additional runs will reduce the variance and we aim to select the locations of these additional runs to maximize the PCS in (5).

We could estimate  $\sigma^2$  from our least squares results and could calculate  $\tilde{\beta}$  using (14) or  $\tilde{y}(x_i)$  using (15). We could then use Monte Carlo simulation with (13) in order to estimate the PCS (Law and Kelton 2000). However, estimating the PCS via Monte Carlo simulation can be time consuming. The next subsection reduces the number of comparisons required and Chapter 3 presents a way to approximate the PCS without running Monte Carlo simulations.

### 2.3. Reducing the Number of Comparisons

Subsections 2.1 and 2.2 demonstrated how we can utilize the quadratic structure of the underlying function in order to provide estimates of the performance measure across the entire domain. This subsection demonstrates that we can also use this quadratic structure to reduce the number of comparisons required in the PCS Equation (5). Specifically, given the discrete domain presented in our problem, we only need to consider the three cases presented in Theorem 1 below.



**Theorem 1:** Given  $x_b$  estimated from the second order polynomial metamodel results, the assumption that our underlying function is quadratic means that we can reduce the required number of comparisons in our *PCS* equations from the  $k - 1$  comparisons expressed in (5) to 2 comparisons. As such, we can restate the OSD problem in (5) as shown in (17) below or in its equivalent form using the parameters  $\tilde{\beta}$ .

$$\begin{aligned} \max_{N_1, \dots, N_k} \quad & PCS = P\{ (\tilde{y}(x_A) \geq \tilde{y}(x_b)) \cap (\tilde{y}(x_Z) \geq \tilde{y}(x_b)) \} \\ \text{s.t.} \quad & N_1 + N_2 + \dots + N_k = T \end{aligned} \quad \text{with} \quad (17)$$

Case 1 (Interior Design Case):  $b \neq 1, k$ ;  $A = b - 1$ ;  $Z = b + 1$ ,

Case 2 (Left Boundary Design Case):  $b = 1$ ;  $A = 2$ ;  $Z = k$ , and

Case 3 (Right Boundary Design Case):  $b = k$ ;  $A = 1$ ;  $Z = k - 1$ .

Proof: Using the definition of conditional probability, we know that

$$\begin{aligned} P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) \mid \forall i\} = \\ P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) ; i = A, Z\} \cdot P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) \mid \forall i \mid \tilde{y}(x_i) \geq \tilde{y}(x_b) ; i = A, Z\} \end{aligned}$$

We aim to show that  $P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) \mid \forall i \mid \tilde{y}(x_i) \geq \tilde{y}(x_b) ; i = A, Z\} = 1$  and must consider the three cases listed above.

Case 1 (Interior Design Case):  $b \neq 1, k$ ;  $A = b - 1$ ;  $Z = b + 1$

For the interior design case, we will show that we know that we have correctly selected if the design that we selected is better than both of its neighboring designs.

In other words, if  $x_b$  is an interior point ( $b \neq 1, k$ ) with  $\tilde{y}(x_{b-1}) \geq \tilde{y}(x_b)$  and

$\tilde{y}(x_{b+1}) \geq \tilde{y}(x_b)$ , then  $\tilde{y}(x_i) \geq \tilde{y}(x_b) \forall i$ . Given that  $\tilde{y}(x_{b-1}) \geq \tilde{y}(x_b)$  and

$\tilde{y}(x_{b+1}) \geq \tilde{y}(x_b)$ , we know that  $(x_{b-1} - x_b)[\tilde{\beta}_1 + \tilde{\beta}_2(x_{b-1} + x_b)] \geq 0$  and

$(x_{b+1} - x_b)[\tilde{\beta}_1 + \tilde{\beta}_2(x_{b+1} + x_b)] \geq 0$ .

Using these equations it can be shown that  $\tilde{\beta}_2 \geq 0$  when  $\tilde{y}(x_{b-1}) \geq \tilde{y}(x_b)$  and

$\tilde{y}(x_{b+1}) \geq \tilde{y}(x_b)$ .

For  $i > 1$ ,  $(x_{b+i} - x_b) \geq (x_{b+1} - x_b)$  and  $(x_{b+i} + x_b) \geq (x_{b+1} + x_b)$ . Therefore, with  $\tilde{\beta}_2 \geq 0$ ,

$(x_{b+i} - x_b)[\tilde{\beta}_1 + \tilde{\beta}_2(x_{b+i} + x_b)] \geq 0$  or  $\tilde{y}(x_{b+i}) \geq \tilde{y}(x_b)$ .

In the same manner, for  $i > 1$ ,  $(x_{b-i} - x_b) \leq (x_{b-1} - x_b)$  and  $(x_{b-i} + x_b) \leq (x_{b-1} + x_b)$ .

Therefore, with  $\tilde{\beta}_2 \geq 0$ ,  $(x_{b-i} - x_b)[\tilde{\beta}_1 + \tilde{\beta}_2(x_{b-i} + x_b)] \geq 0$  or  $\tilde{y}(x_{b-i}) \geq \tilde{y}(x_b)$ . These

results provide the fact that if  $x_b$  is an interior point,

$$P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) \forall i \mid \tilde{y}(x_i) \geq \tilde{y}(x_b); i = b-1, b+1\} = 1$$

Case 2 (Left Boundary Design Case):  $b = 1$ ;  $A = 2$ ;  $Z = k$

For the left boundary design case, our objective function also simplifies to two comparisons. We will show that we know that we have correctly selected if the design that we selected is better than both the adjacent design and the opposite boundary design.

For the left boundary, if we are given that  $\tilde{y}(x_2) \geq \tilde{y}(x_b)$  and  $\tilde{y}(x_k) \geq \tilde{y}(x_b)$ , then it can

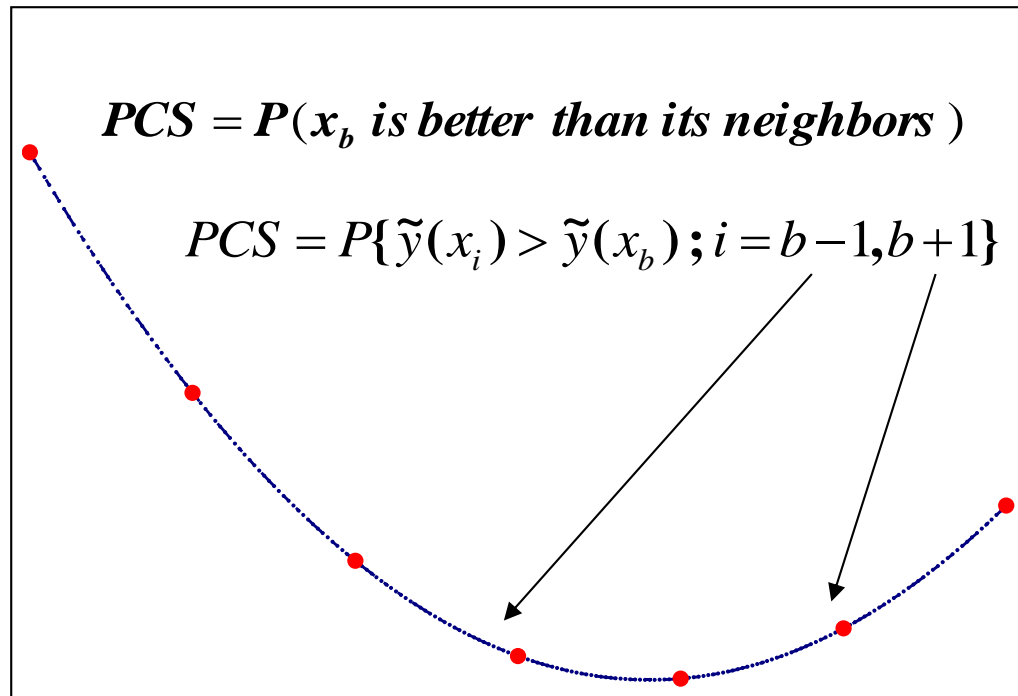
be shown that  $\tilde{y}(x_i) \geq \tilde{y}(x_b) \forall i$ . To start, we define  $g(x_i) = [\tilde{\beta}_1 + \tilde{\beta}_2(x_i + x_b)]$ . Since

$(x_i - x_b) > 0 \forall i \neq b$  and we are given that  $\tilde{y}(x_2) \geq \tilde{y}(x_b)$  and  $\tilde{y}(x_k) \geq \tilde{y}(x_b)$ , we know that  $g(x_2) \geq 0$  and  $g(x_k) \geq 0$ . Since  $g(x_i)$  is a linear function and the two extreme points have values greater than or equal to zero, we can establish that  $[\tilde{\beta}_1 + \tilde{\beta}_2(x_i + x_b)] \geq 0 \forall i \neq b$ . Using again that  $(x_i - x_b) > 0 \forall i \neq b$ , we know that  $\tilde{y}(x_i) \geq \tilde{y}(x_b) \forall i \neq b$ . These results provide the fact that if  $x_b = x_1$  then  $P\{\tilde{y}(x_i) \geq \tilde{y}(x_b) \forall i \mid \tilde{y}(x_i) \geq \tilde{y}(x_b); i = 2, k\} = 1$ .

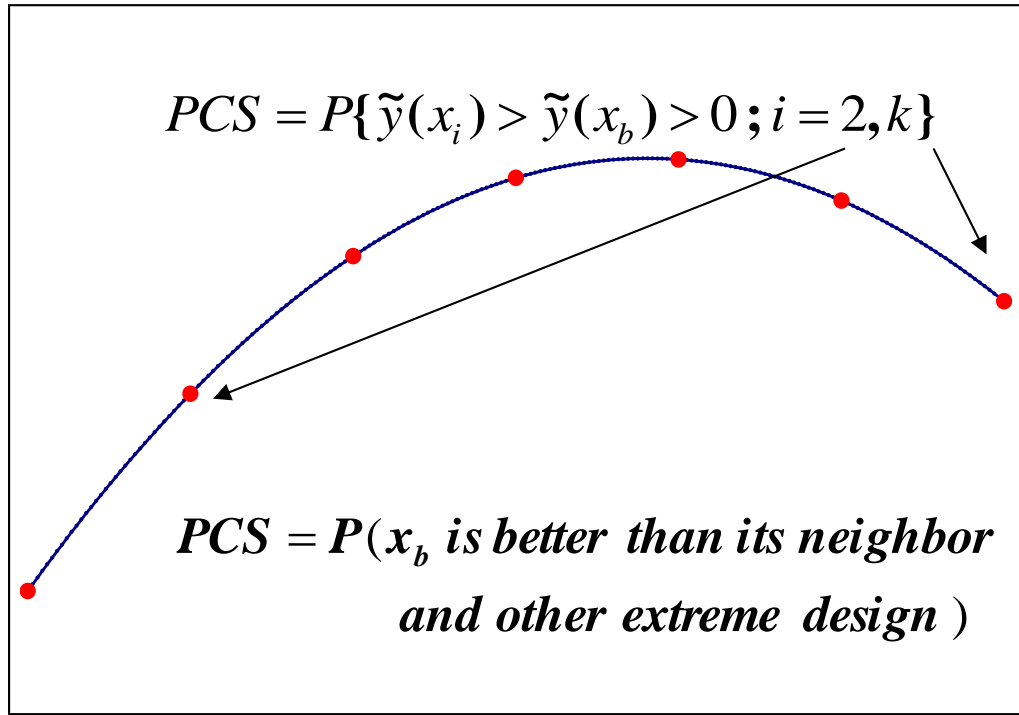
Case 3 (Right Boundary Design Case):  $b = k$ ;  $A = 1$ ;  $Z = k - 1$ .

The proof for the right boundary design case where  $x_b = x_k$  is almost identical to the approach presented for the left boundary design case except that  $(x_i - x_b) < 0$  and  $[\tilde{\beta}_1 + \tilde{\beta}_2(x_i + x_b)] \leq 0 \forall i \neq b$ .

Figure 2 below illustrates the results for the interior case and Figure 3 below provides one depiction of a boundary case.



**Figure 2: Two Comparisons (Interior Case)**



**Figure 3: Two Comparisons (Boundary Case)**

## CHAPTER 3 OPTIMAL SIMULATION DESIGN METHOD

In Chapter 2, we provided a derivation for our Bayesian regression model and showed that we required only two comparisons by using the information from the regression equation. These results were generally independent of how we allocated the simulation budget to the design locations. In this section, we will develop the OSD allocation method. We begin by deriving an approximation to our PCS equation. We then establish that it is sufficient to allocate simulation runs to only three design locations to estimate the parameters for the quadratic function. (For notation sake, we will refer to the three design locations receiving simulation runs as support points.) We then derive an optimal allocation of simulation runs to the three support points and an efficient approximation for the optimal allocation. Finally, we determine the optimal locations for the three support points.

### 3.1. A Lower Bound for PCS

Given the comparisons from the PCS equation in (17), we can use the complementary conditions for probabilistic events to establish that

$$P\{(\tilde{y}(\mathbf{x}_A) \geq \tilde{y}(\mathbf{x}_b)) \cap (\tilde{y}(\mathbf{x}_Z) \geq \tilde{y}(\mathbf{x}_b))\} = 1 - P\{(\tilde{y}(\mathbf{x}_A) \leq \tilde{y}(\mathbf{x}_b)) \cup (\tilde{y}(\mathbf{x}_Z) \leq \tilde{y}(\mathbf{x}_b))\}.$$

In addition, we know from probability theory that

$$\begin{aligned}
& P\{(\tilde{y}(x_A) \leq \tilde{y}(x_b)) \cup (\tilde{y}(x_Z) \leq \tilde{y}(x_b))\} = \\
& P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} + P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\} - P\{(\tilde{y}(x_A) \leq \tilde{y}(x_b)) \cap (\tilde{y}(x_Z) \leq \tilde{y}(x_b))\}
\end{aligned}$$

Substituting this result, we obtain

$$\begin{aligned}
& PCS = \\
& 1 - P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} - P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\} + P\{(\tilde{y}(x_A) \leq \tilde{y}(x_b)) \cap (\tilde{y}(x_Z) \leq \tilde{y}(x_b))\}.
\end{aligned}$$

Using the results from Chapter 2, the last term in the equation above determines the probability that  $x_b$  is actually the “worst” design in the domain such that it is the one with the largest expected performance measure. This probability is typically small so that we can establish a lower bound (or approximate PCS) for our PCS equation as shown in (18) below.

$$PCS \geq APCS = 1 - P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} - P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\}. \quad (18)$$

### 3.2. Sufficient Number of Support Points

In this section, we will establish a relationship between the information matrix commonly used in the DOE literature and our PCS criterion in order to leverage results previously established in the DOE literature. Define

$$\tilde{d}(x_i) \equiv \tilde{y}(x_i) - \tilde{y}(x_b) = \tilde{\beta}_2(x_i^2 - x_b^2) + \tilde{\beta}_1(x_i - x_b).$$

This result shows that  $\tilde{d}(x_i)$  is a linear combination of the  $\tilde{\beta}$  elements so the  $\tilde{d}(x_i)$

terms are also normally distributed. Using the results of Section 2,  $\tilde{d}(x_i) \sim N[\hat{d}(x_i), \varsigma_i]$

where

$$\hat{d}(x_i) = \hat{y}(x_i) - \hat{y}(x_b)$$

and

$$\varsigma_i = \sigma^2 \begin{pmatrix} 0, & x_i - x_b, & x_i^2 - x_b^2 \end{pmatrix} (X^t X)^{-1} \begin{pmatrix} 0 \\ x_i - x_b \\ x_i^2 - x_b^2 \end{pmatrix}. \quad (19)$$

Based upon the fact that  $\tilde{d}(x_i) \sim N[\hat{d}(x_i), \varsigma_i]$  we know that

$$P\{\tilde{y}(x_i) \leq \tilde{y}(x_b)\} = P\{-\tilde{d}(x_i) \geq 0\} = \int_0^\infty \frac{1}{\sqrt{2\pi\varsigma_i}} e^{\frac{-[v - (-\hat{d}(x_i))]^2}{2\varsigma_i}} dv.$$

Conducting a change of variables such that  $r = \frac{v + \hat{d}(x_i)}{\sqrt{\varsigma_i}}$  and  $dv = \sqrt{\varsigma_i} dr$ ,

$$P\{-\tilde{d}(x_i) \geq 0\} = \int_{\frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}}}^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-r^2}{2}} dr. \quad (20)$$

Equation (20) shows that the comparisons in (18) are a function of  $\hat{d}(x_i)$  and  $\varsigma_i$ . As we take additional simulation runs,  $E[\tilde{d}(x_i)] = \hat{d}(x_i)$  and (19) shows that  $\varsigma_i$  is a function of the information matrix  $(X^t X)$ . Therefore, the information obtained by additional simulation runs will affect  $\varsigma_i$  and thus our PCS criterion.

For a particular support point and allocation scheme  $\gamma$ , denote  $(X^t X)_\gamma$  as the resulting information matrix. By definition, the information matrix  $(X^t X)_1$  dominates the information matrix  $(X^t X)_2$  or  $(X^t X)_1 \geq (X^t X)_2$  when  $(X^t X)_1 - (X^t X)_2$  results in a



non-negative definite matrix. A criterion  $C$  conforms with a Loewner ordering if

$(X'X)_1 \geq (X'X)_2$  then  $C_1 \geq C_2$  or, to simplify the notation,

$$(X'X)_1 \geq (X'X)_2 \Rightarrow C_1 \geq C_2 \text{ (Liski, et al., 2001).}$$

**Lemma 1:** The PCS criterion conforms with a Loewner ordering such that

$$(X'X)_1 \geq (X'X)_2 \Rightarrow APCS_1 \geq APCS_2.$$

Proof: This proof is similar to one presented by Ehrenfeld (1955). It is well established in

the DOE literature that  $(X'X)_1 \geq (X'X)_2 \Rightarrow (\varsigma_i)_1 \leq (\varsigma_i)_2$  (see Atkinson and Donev,

1998). Using this result, we can establish that

$$(X'X)_1 \geq (X'X)_2 \Rightarrow \left( \frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}} \right)_1 \geq \left( \frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}} \right)_2.$$

Therefore, using (20) we can also establish that

$$(X'X)_1 \geq (X'X)_2 \Rightarrow (P\{-\tilde{d}(x_i) \geq 0\})_1 \leq (P\{-\tilde{d}(x_i) \geq 0\})_2.$$

or

$$(X'X)_1 \geq (X'X)_2 \Rightarrow (P\{\tilde{y}(x_i) \geq \tilde{y}(x_b)\})_1 \leq (P\{\tilde{y}(x_i) \geq \tilde{y}(x_b)\})_2.$$

Using these results with our APCS equation in (18), we obtain that

$$(X'X)_1 \geq (X'X)_2 \Rightarrow APCS_1 \geq APCS_2.$$

Note that establishing that the PCS criterion conforms to a Loewner ordering in Lemma 1 does not imply that we can substitute the information matrix for our PCS criterion. Instead, it allows us to reduce the set of possible allocation combinations. We offer Lemma 2 as a simple example of this.

**Lemma 2:** Given a simulation budget of  $T$ , we will maximize the PCS by utilizing the entire simulation budget.

Proof: Let information matrix  $(X^T X)_\Psi$  denote an information matrix using  $\Psi$  simulation runs such that  $\Psi < T$ . Let  $(X^T X)_{\Psi+1}$  denote the information matrix after allocating one additional simulation run. Using the results from Section 2,

$$(X^T X)_\Psi = \begin{bmatrix} \Psi & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}$$

$$(X^T X)_{\Psi+1} - (X^T X)_\Psi = \begin{bmatrix} 1 & x_{\Psi+1} & x_{\Psi+1}^2 \\ x_{\Psi+1} & x_{\Psi+1}^2 & x_{\Psi+1}^3 \\ x_{\Psi+1}^2 & x_{\Psi+1}^3 & x_{\Psi+1}^4 \end{bmatrix}.$$

Regardless of the location of the additional simulation run, it is shown that

$(X^T X)_{\Psi+1} \geq (X^T X)_\Psi$ , so it is better to allocate one additional simulation run, and by extension better to allocate the entire simulation budget. We have therefore reduced the possible allocation combinations to only the possible combinations that allocate the entire simulation budget.

However, the results in Lemma 2 do not determine where to allocate each additional simulation run given the choice of  $k$  design locations. The following theorem based upon the DOE literature will at least allow us to reduce the number of support points required for our allocations.

**Theorem 2:** Given that we assume the expectation of our underlying function is quadratic, we require only three support points to obtain all of the information in the  $X^T X$  matrix. Two of these support points will be at the extreme design locations ( $x_1$  and  $x_k$ ).

Proof: de la Garza (1954) establishes that for a polynomial of degree  $m$  and a discrete domain with more than  $m+1$  support points, the information obtained in the  $X^T X$  matrix by a spacing at more than  $m+1$  support points can always be attained by spacing the same information at only  $m+1$  of the design locations. Kiefer (1959) extends this result by proving that regardless of the optimality criteria, for a polynomial with  $m+1$  support points, two of the support points are at the extremes.

Given the results of Theorem 2, we will refer to the support points as  $\{x_1, x_s, x_k\}$  where  $x_1 < x_s < x_k$ . Using this notation and the APCS in (18), we can now restate the OSD problem in (17) as the OSD problem in (21) below.

#### OSD Problem

$$\begin{aligned}
 \max_{N_1, N_s, N_k} \quad & APCS = 1 - P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} - P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\} \\
 \text{s.t.} \quad & N_1 + N_s + N_k = T; \\
 & N_i = 0 \forall i \neq 1, s, k
 \end{aligned}
 \quad \text{with} \quad (21)$$

Case 1 (Interior Design Case):  $b \neq 1, k$ ;  $A = b - 1$ ;  $Z = b + 1$ ,

Case 2 (Left Boundary Design Case):  $b = 1$ ;  $A = 2$ ;  $Z = k$ , and

Case 3 (Right Boundary Design Case):  $b = k$ ;  $A = 1$ ;  $Z = k - 1$ .

Having established the relationship between the information matrix commonly used in the DOE literature and our PCS criterion, we were able to use the results from the DOE literature (de la Garza, 1954 and Kiefer, 1959) to reduce the number of support points. However, as noted by Liski, et al. (2001), establishing dominance within the class of three point allocation schemes using only the information matrix is generally not feasible. The optimality criterion will instead guide this determination. As a simple example, for a quadratic underlying function, using the D-optimality criterion will always result in allocating one third of the simulation budget to both extreme points of the domain and the remaining third to a support point located at the center of the domain. Using a DS-optimality criterion that is interested in only the  $\beta_1$  and  $\beta_2$  parameters will result in allocating one half of the simulation budget to both extreme points of the domain.

### 3.3. Optimal Allocations

In this section we will first develop a method to optimally allocate simulation runs to the three support points  $\{x_1, x_s, x_k\}$ . In the next two subsections, we will use the results to determine an efficient approximation for the optimal allocation and the optimal design location for the support point  $x_s$ . Since our aim is to efficiently allocate the computing

budget to the three support points, we will rewrite the variance term in (19) so that it is expressed in terms of the percentage of the simulation budget allocated to each support point. For notation sake, we define  $\alpha_j = N_j / T$ ,  $j \in \{1, s, k\}$ .

We begin by examining the symmetric nature of the  $X'X$  matrix. Given the fact that we are only taking simulation replications at the three support points, this matrix can be rewritten in terms of the support points as

$$X'X = T \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_s & x_k \\ x_1^2 & x_s^2 & x_k^2 \end{pmatrix} \begin{pmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_s & 0 \\ 0 & 0 & \alpha_k \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_s & x_s^2 \\ 1 & x_k & x_k^2 \end{pmatrix}.$$

Using a basic property for matrices where  $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$  (Neter et al., 1996), we obtain

$$(X'X)^{-1} = \frac{1}{T} \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_s & x_s^2 \\ 1 & x_k & x_k^2 \end{pmatrix}^{-1} \begin{pmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_s & 0 \\ 0 & 0 & \alpha_k \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_s & x_k \\ x_1^2 & x_s^2 & x_k^2 \end{pmatrix}^{-1}.$$

It can be shown that

$$\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_s & x_k \\ x_1^2 & x_s^2 & x_k^2 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{x_s x_k}{(x_1 - x_s)(x_1 - x_k)} & \frac{-(x_s + x_k)}{(x_1 - x_s)(x_1 - x_k)} & \frac{1}{(x_1 - x_s)(x_1 - x_k)} \\ \frac{x_1 x_k}{(x_s - x_1)(x_s - x_k)} & \frac{-(x_1 + x_k)}{(x_s - x_1)(x_s - x_k)} & \frac{1}{(x_s - x_1)(x_s - x_k)} \\ \frac{x_1 x_s}{(x_k - x_1)(x_k - x_s)} & \frac{-(x_1 + x_s)}{(x_k - x_1)(x_k - x_s)} & \frac{1}{(x_k - x_1)(x_k - x_s)} \end{pmatrix}. \quad (22)$$

Therefore, after computing the inverse in (22), we may write (19) in the simplified form:

$$\varsigma_i = \frac{\sigma^2}{T} \left[ \frac{D_{i,1}^2}{\alpha_1} + \frac{D_{i,s}^2}{\alpha_s} + \frac{D_{i,k}^2}{\alpha_k} \right],$$

where

$$D_{i,1} = \left\{ \frac{(x_s - x_i)(x_k - x_i) - (x_s - x_b)(x_k - x_b)}{(x_1 - x_s)(x_1 - x_k)} \right\},$$

$$D_{i,s} = \left\{ \frac{(x_1 - x_i)(x_k - x_i) - (x_1 - x_b)(x_k - x_b)}{(x_s - x_1)(x_s - x_k)} \right\},$$

$$D_{i,k} = \left\{ \frac{(x_1 - x_i)(x_s - x_i) - (x_1 - x_b)(x_s - x_b)}{(x_k - x_1)(x_k - x_s)} \right\}. \quad (23)$$

Note that  $D_{i,1}$ ,  $D_{i,s}$ , and  $D_{i,k}$  are the Lagrange interpolating polynomial coefficients (see de la Garza, 1954 and Burden and Faires, 1993) for estimating  $\hat{d}(x_i)$  at the three support points  $\{x_1, x_s, x_k\}$ . Defining  $\bar{y}(x_j) = f(x_j)/N_j$ ,  $j \in \{1, s, k\}$  as the mean of the simulation runs conducted at the support point  $x_j$  and noting the interpolating polynomial property at the support points such that  $\hat{y}(x_j) = \bar{y}(x_j)$ , we see that

$$\hat{d}(x_i) = D_{i,1}\bar{y}(x_1) + D_{i,s}\bar{y}(x_s) + D_{i,k}\bar{y}(x_k) = D_{i,1}\hat{y}(x_1) + D_{i,s}\hat{y}(x_s) + D_{i,k}\hat{y}(x_k). \quad (24)$$

Given the unbiased properties of our least squares parameters and estimators, we assume that there will not be large changes in  $\hat{d}(x_i)$  as we take additional simulation runs. On the other hand, we note from (23) that  $\varsigma_i$  and thus the APCS expressed in (21) are dependent upon  $T$  and the allocations to each of the three support points. Theorem 3 below provides an optimal allocation rule for the OSD problem.

**Theorem 3:** The OSD problem given in (21) can be maximized with the following allocation rule:

$$\alpha_i = \frac{\sqrt{q_i}}{\sqrt{q_1} + \sqrt{q_s} + \sqrt{q_k}}, i = 1, s, k$$

$$\alpha_i = 0, i \neq 1, s, k$$
(25)

where  $q_j = \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}}$  and  $\phi(r) = \frac{1}{\sqrt{2\pi}} e^{-\frac{r^2}{2}}$ .

Proof:

To solve the OSD problem given in (21), we define a Lagrangian function

$$Q = APCS - \lambda \left( \sum_{i=1}^k \alpha_i - 1 \right) \text{ or using (20)}$$

$$Q = 1 - \int_{\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}}^{\infty} \phi(r) dr - \int_{\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}}^{\infty} \phi(r) dr - \lambda \left( \sum_{i=1}^k \alpha_i - 1 \right)$$
(26)

so that we can investigate  $\frac{\partial Q}{\partial \alpha_j}$  to determine the allocation. We can use the chain rule to

establish that

$$\frac{\partial Q}{\partial \alpha_j} = - \frac{\partial Q}{\partial \varsigma_A} \frac{\partial \varsigma_A}{\partial \alpha_j} - \frac{\partial Q}{\partial \varsigma_Z} \frac{\partial \varsigma_Z}{\partial \alpha_j} - \lambda.$$

From (26) we obtain

$$\frac{\partial Q}{\partial \varsigma_i} = \phi\left(\frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}}\right) \frac{\hat{d}(x_i)}{2(\varsigma_i)^{3/2}}.$$
(27)

From (23) we also obtain

$$\frac{\partial \varsigma_i}{\partial \alpha_j} = \frac{-\sigma^2}{T} \frac{D_{i,j}^2}{\alpha_j^2}. \quad (28)$$

Substituting these results we determine that the partial derivative of our objective function with respect to the allocation at each of our support points is

$$\frac{\partial Q}{\partial \alpha_j} = \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A)}{2(\varsigma_A)^{3/2}} \frac{\sigma^2}{T} \frac{D_{A,j}^2}{\alpha_j^2} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z)}{2(\varsigma_Z)^{3/2}} \frac{\sigma^2}{T} \frac{D_{Z,j}^2}{\alpha_j^2} - \lambda. \quad (29)$$

By setting  $\frac{\partial Q}{\partial \alpha_j} = 0$ , we obtain

$$\frac{1}{\alpha_j^2} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] = \frac{2\lambda T}{\sigma^2}.$$

This yields

$$\frac{\sqrt{q_1}}{\alpha_1} = \frac{\sqrt{q_s}}{\alpha_s} = \frac{\sqrt{q_k}}{\alpha_k} \quad (30)$$

with

$$q_j = \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}}.$$

Using the fact that  $\alpha_1 + \alpha_s + \alpha_k = 1$  and (30), we can establish that

$$\alpha_i = \frac{\sqrt{q_i}}{\sqrt{q_1} + \sqrt{q_s} + \sqrt{q_k}}, i = 1, s, k.$$



**Theorem 4:** The objective function (26) is concave. Therefore, the allocation rule given in (25) will yield a global (although not necessarily unique) optimal solution to (26).

Proof: See Appendix A.

While the allocation rule given in (25) is nonlinear with respect to  $\alpha$ , the solution for each allocation as depicted in (25) or the alternate form represented by (30) shows that the allocations are contained within a simplex generated by the constraint  $\alpha_1 + \alpha_s + \alpha_k = 1$ . Since this simplex represents a compact subset of values for  $\alpha$ , we know there exists a fixed point solution for the system of equations in (25) (Burden and Faires, 1993) and can use one of several root finding techniques to find a solution. For guaranteed convergence, we can use the steepest descent method to find a solution to the system of equations in (25) by finding a minimum to (31) below (Burden and Faires, 1993).

$$\Gamma = \left( \alpha_1 - \frac{\sqrt{q_1}}{\sqrt{q_1} + \sqrt{q_s} + \sqrt{q_k}} \right)^2 + \left( \alpha_s - \frac{\sqrt{q_s}}{\sqrt{q_1} + \sqrt{q_s} + \sqrt{q_k}} \right)^2 + \left( \alpha_k - \frac{\sqrt{q_k}}{\sqrt{q_1} + \sqrt{q_s} + \sqrt{q_k}} \right)^2 \quad (31)$$

### 3.4. Approximately Optimal Allocations

In the previous section we derived an optimal allocation rule for the OSD problem in (21). However, our aim is to provide a simple means for efficiently allocating the

budget instead of numerically computing the optimal values for  $\alpha$  as expressed in (25).

In this section, we will derive an approximation for our optimal allocations and our numerical results in Section 4 will demonstrate its efficiency.

Define design  $x_M$  such that  $M = \arg \max_{i=\Lambda, Z} \left\{ \phi\left(\frac{\hat{d}(x_i)}{\sqrt{\zeta_i}}\right) \right\} = \arg \min_{i=\Lambda, Z} \left\{ \frac{\hat{d}(x_i)}{\sqrt{\zeta_i}} \right\}$ . Revisiting the OSD

problem in (21), we note the following lower and upper bounds for our APCS equation as shown in (32) below:

$$1 - 2P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\} \leq APCS \leq 1 - P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\} \quad (32)$$

If we denote  $L$  and  $U$  as the lower and upper bounds respectively, we obtain:

$$\frac{\partial L}{\partial \alpha_j} = 2\phi\left(\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}\right) \frac{\hat{d}(x_M)}{2(\zeta_M)^{3/2}} \frac{\sigma^2}{T} \frac{D_{M,j}^2}{\alpha_j^2} - \lambda = 0$$

$$\frac{\partial U}{\partial \alpha_j} = \phi\left(\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}\right) \frac{\hat{d}(x_M)}{2(\zeta_M)^{3/2}} \frac{\sigma^2}{T} \frac{D_{M,j}^2}{\alpha_j^2} - \lambda = 0$$

Each of these bounds results in the approximately optimal allocation of

$$\alpha_i = \frac{|D_{M,i}|}{|D_{M,1}| + |D_{M,s}| + |D_{M,k}|}, i = 1, s, k. \quad (33)$$

$$\alpha_i = 0, i \neq 1, s, k$$

Using the concepts from the research area of large deviation theory, we know that our lower and upper bounds will have the same asymptotic convergence rate since they are based upon the same allocation rule (33). More importantly, since our APCS is bounded by them, it will also converge at the same rate. For brevity, we omit a detailed discussion

of the large deviation theory and reference Dembo and Zeitouni (1992) for a general discussion and Glynn and Juneja (2004) for a simulation optimization application.

### 3.5. Optimal Support Point Location

Given the approximately optimal allocations to  $\{x_1, x_s, x_k\}$  from the previous section, we will determine the optimal design location for the support point  $x_s$  among all the given design points. Without loss of any generality but for clarity and ease in notation, we present the following theorem assuming an odd number of designs evenly spaced across the domain. We then address cases with uneven spacing or an even number of designs.

**Theorem 5:** When presented with an odd number of design locations evenly spaced

across the domain such that  $\Delta = x_{i+1} - x_i = \frac{x_k - x_1}{k-1} \forall i = 1, \dots, k-1$ , the approximate PCS

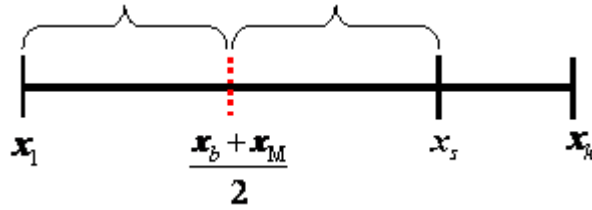
expressed in (18) with allocations that satisfy (25) is maximized with the following

locations for  $x_s$ :

$$\mathbf{x}_s = \begin{cases} \mathbf{x}_{M+b-1}, & \frac{3\mathbf{x}_1 + \mathbf{x}_k}{4} \leq \frac{\mathbf{x}_M + \mathbf{x}_b}{2} < \frac{\mathbf{x}_1 + \mathbf{x}_k}{2} \\ \mathbf{x}_{M+b-k}, & \frac{\mathbf{x}_1 + \mathbf{x}_k}{2} < \frac{\mathbf{x}_M + \mathbf{x}_b}{2} \leq \frac{\mathbf{x}_1 + 3\mathbf{x}_k}{4} \\ \mathbf{x}_{(k+1)/2}, & \text{otherwise} \end{cases} \quad (34)$$

Proof: In Appendix B, we consider five cases and analyze  $\frac{\partial Q}{\partial x_s}$  for each case.

The results of Theorem 5 demonstrate that when  $(x_M + x_b)/2$  is in the middle half of the domain,  $x_s$  is located such that it is the same distance from  $(x_M + x_b)/2$  as the nearest boundary is from  $(x_M + x_b)/2$ . This makes the design symmetric about  $(x_M + x_b)/2$ . The first case is illustrated in Figure 4 below.



**Figure 4: Location of  $x_s$  for the first case in Theorem 5**

When  $(x_M + x_b)/2$  is in the outer half of the domain,  $x_s$  is located at the center of the domain making the design symmetric about  $x_s$ .

If the design locations are not evenly spaced or if we have an even number of design locations, we can use a subgradient approach similar to the approach taken for Theorem 5. The results of this type of approach provide that the approximate PCS expressed in (18) with allocations that satisfy (25) is maximized by selecting the interior design location closest to  $x_s$  selected from Theorem 5.

### 3.6. OSD Procedure

The following is the algorithm that we used to implement the OSD method for the experiments. The actual ProModel code used is in Appendix C.

### **OSD Procedure (Maximizing PCS)**

**INPUT**         $k$  (the number of design locations),  $T$  (the computing budget),  $x_i$  (the design locations),  $n_0$  (the number of initial runs),  $\theta_j$  (the number runs allocated each iteration  $j$ );

**INITIALIZE**  $j \leftarrow 0$ ;

Perform  $n_0$  simulation replications for three design locations; by convention we use the D-opt support points such that  $N_1^j = N_{(k+1)/2}^j = N_k^j = n_0$ .

**LOOP WHILE**  $\sum_{i=1}^k N_i^j < T$  **DO**

**UPDATE**    - Estimate a quadratic regression equation using the information from all prior simulation runs.

- Estimate the mean and variance of each design location using

$$\hat{y}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2.$$

- Determine the observed best design so that  $x_b = \underset{i}{\mathbf{arg\,min}} \hat{y}(x_i)$ .

- Based upon the location of  $x_b$ , use (21) to determine  $x_A$  and  $x_Z$ .

- Determine  $x_M$  such that  $M = \underset{i=A,Z}{\mathbf{arg\,max}} \left\{ \phi\left(\frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}}\right) \right\} = \underset{i=A,Z}{\mathbf{arg\,min}} \left\{ \frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}} \right\}$ .

- Determine  $x_s$  using (34).

**ALLOCATE** Increase the computing budget by  $\theta_{j+1}$  and calculate the new budget allocations for  $\alpha_1^{j+1}$ ,  $\alpha_s^{j+1}$  and  $\alpha_k^{j+1}$  according to (33).

**SIMULATE** Perform  $\text{round}[\theta_{j+1}\alpha_i^{j+1}]$  simulations for design  $i, i = 1, s, k; j \leftarrow j+1$ .

**END OF LOOP**

### 3.7. Numerical Testing and Results

In this section, we describe how we compared the results from our new OSD method against the results from four other allocation procedures. We start by providing a description of the other methods chosen to provide a perspective of the efficiency gained by using the approximately optimal allocations and also by using the information from a regression equation. We then describe our testing framework. Finally, we end with the results from six experiments.

The simplest allocation case is a naive method that equally allocates (EA) the runs to each design location such that  $N_i = T/k$  for each  $i$ . For this method, we designate the design location with the smallest mean performance measure as  $x_b$  so that

$$x_b = \underset{i}{\text{arg min}} \frac{\sum_{j=1}^{T/k} f(x_i)}{T/k}.$$

Instead of equally allocating, we also tested the OCBA method, which is one of efficient R&S performers (Branke et al. 2005, 2007). This method requires a set of initialization runs and, based upon the findings of Chen et al. (2000), we used an initial allocation of 5 runs for each design location. For the type of problem that we are investigating with an assumed underlying quadratic function and homoscedastic variance,

this method will allocate most of the simulation budget to the best design and its immediate neighbors.

The next method in our progression equally allocates to each design location but uses a response surface (EA-RS) to compare the results. As with the OSD method, we designate  $x_b$  so that  $x_b = \underset{i}{\mathbf{arg\,min}} \hat{y}(x_i)$ .

The final method that we will compare against leverages the results from de la Garza (1954) in which we require only three support points to capture all of the information in the response. A very popular way to do this in the DOE literature is to use the D-optimality criterion (D-opt) that maximizes the determinant of the information matrix resulting in minimizing the generalized variance of the parameter estimates. Liski et al. (2001) provide an overview of the development of this popular method, its many extensions, and its relationship with other common methods. Atkinson and Donev (1998) provide a list of properties of this criterion and note that D-opt often performs well compared to other criteria. For an underlying quadratic function, this criterion establishes support points at the two extreme points and at the center of the domain and allocates one third of the simulation budget to each of these support points. Using the notation from our OSD derivation, this criterion will always allocate with  $\{\alpha_1, \alpha_{(k+1)/2}, \alpha_k\} = \{1/3, 1/3, 1/3\}$ .

For the OSD method, we initialize as per the algorithm described in the previous section with  $N_1 = N_{(k+1)/2} = N_k = n_0 = 2$ .

In order to compare the results of using OSD against these other methods, we conducted the first five experiments using the following function to represent the simulation output:

$$f(x_i) = (x_i - a)^2 + N(0,1) .$$

We used a domain consisting of evenly spaced design locations where  $x \in [-1, 1]$ .

For the sixth experiment, we used the following function to represent the simulation output:

$$f(x_i) = (x_i - 1)^4 + N(0,1) .$$

We used a domain consisting of evenly spaced design locations where  $x \in [-4, 4]$ . With a slight modification to our OSD procedure, we are able to demonstrate using this last experiment that OSD may be robust enough to handle problems that are not quadratic across the entire domain if we can partition the domain such that each partition is approximately quadratic. This concept will be explored in more detail in Chapter 4 and demonstrated with more general underlying functions.

We conducted the first four experiments using a total computing budget of 1,000 runs, the fifth experiment with 45,000 runs, and the sixth experiment with 2,000 runs.

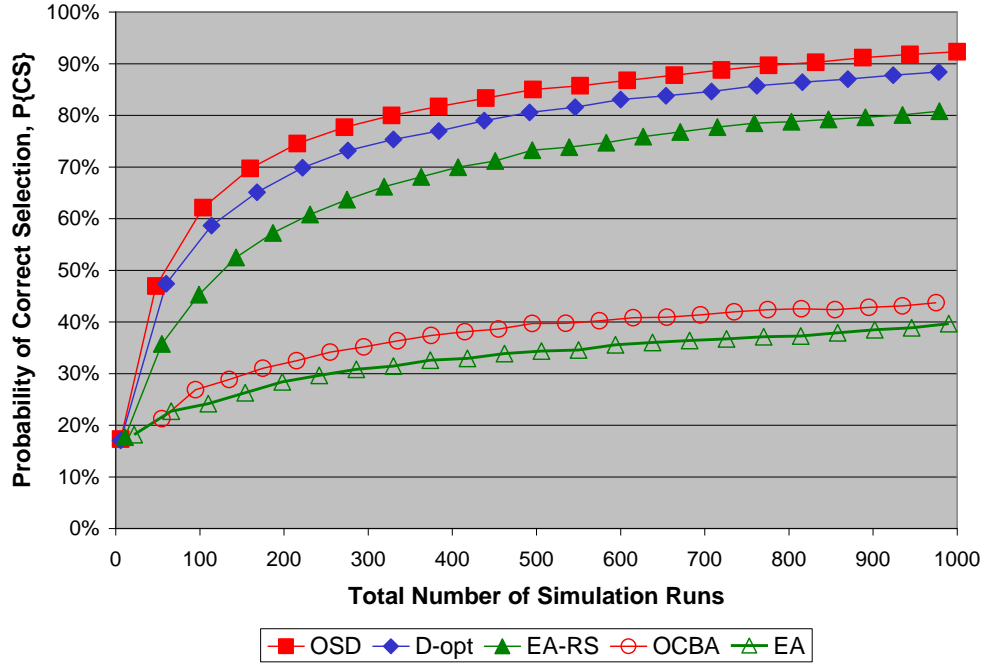
The results will show that these amounts are sufficient to compare the performance of the methods. To mitigate the fact that the methods have varying fixed costs associated with them and in order to compare the performance of the methods using various simulation budgets, we calculate the PCS for each method during each iteration until the total



simulation budget is exhausted. We repeat this whole procedure 10,000 times and then estimate the PCS obtained for each method after these 10,000 independent applications.

### **Experiment 1 ( $a = 1/3$ , 11 design locations)**

The domain consists of 11 design locations where  $x \in [-1, -0.8, \dots, 1]$ . For OSD and OCBA, we provided initial runs as described above and then allocated 8 runs for OCBA and 14 runs for OSD during additional iterations. The other methods provided allocations as described above. Figure 5 shows the simulation results. The methods using a regression equation clearly perform the best. As a point of comparison, we obtain a PCS of 40% after 77 runs with the methods using the regression equations, 551 runs with OCBA, and 1,000 runs with EA. Since OSD initializes with the D-optimal method, there is little initial difference between the two methods. However, after 1,000 runs, D-optimal achieves an 88.8% PCS while OSD achieves the same PCS after only 734 runs. Both of these methods perform significantly better than EA-RS.



**Figure 5: Results of Experiment 1**

Table 1 below provides the resulting OSD allocations for several computing budgets. Since  $a = 1/3$ , the best design location will be  $x_8 = 0.4$ , and so  $x_A = x_7 = 0.2$  and  $x_Z = x_9 = 0.6$ . Asymptotically, the OSD rule will pick  $x_1$ ,  $x_4$ , and  $x_{11}$  as the support points with the allocation ratios  $\{\alpha_1, \alpha_4, \alpha_{11}\} = \{0, 1/2, 1/2\}$ . The results in Table 1 show that when  $T$  increases, the OSD procedure gradually approaches this allocation ratio.

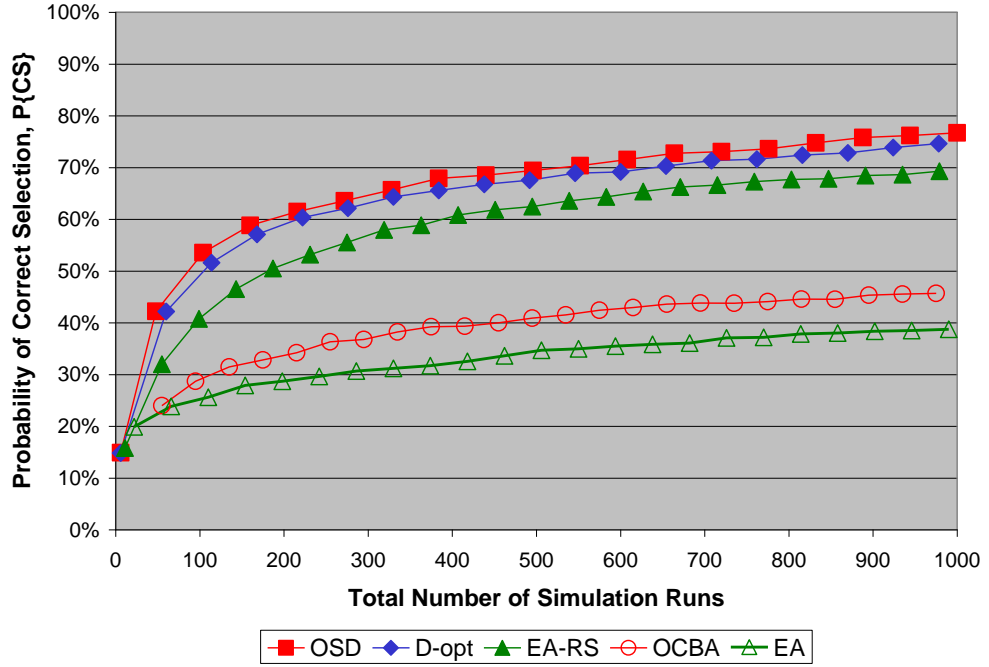
On the other hand, the D-optimal will choose  $x_1$ ,  $x_6$ , and  $x_{11}$  as the support points with allocation ratios  $\{\alpha_1, \alpha_6, \alpha_{11}\} = \{1/3, 1/3, 1/3\}$ . Note that the allocation rules chosen by D-optimal and OSD are very different and so the resulting PCS are very different.

Table 1: OSD Allocations for Experiment 1

Total Runs	Design 1 ( $x = -1$ )	Design 2 ( $x = -0.8$ )	Design 4 ( $x = -0.4$ )	Design 6 ( $x = 0$ )	Design 11 ( $x = 1$ )
160	0.05%	0.64%	38.52%	10.84%	49.95%
440	0.00%	0.20%	46.91%	2.89%	50.00%
720	0.00%	0.14%	49.09%	0.77%	50.00%
1000	0.00%	0.14%	49.63%	0.23%	50.00%

**Experiment 2 ( $a = 2/3$ , 11 design locations)**

Figure 6 presents the simulation results for Experiment 2. The allocation methods are the same as described for Experiment 1 and the results are similar to the results from Experiment 1 with the exception that D-optimal is more competitive with OSD. After 1,000 runs, D-optimal achieves a 74.7% PCS while OSD achieves the same PCS after only 832 runs.



**Figure 6: Results of Experiment 2**

Table 2 below provides the resulting OSD allocations for several computing budgets. For this experiment, the best design location will be  $x_9 = 0.6$ , and so  $x_A = x_8 = 0.4$  and  $x_Z = x_{10} = 0.8$ . Asymptotically, the OSD rule will pick the same support points with D-Optimal, i.e.,  $x_1$ ,  $x_6$ , and  $x_{11}$  but with  $\{\alpha_1, \alpha_6, \alpha_{11}\} = \{1/14, 1/2, 3/7\}$  instead of one third allocated to each support point.

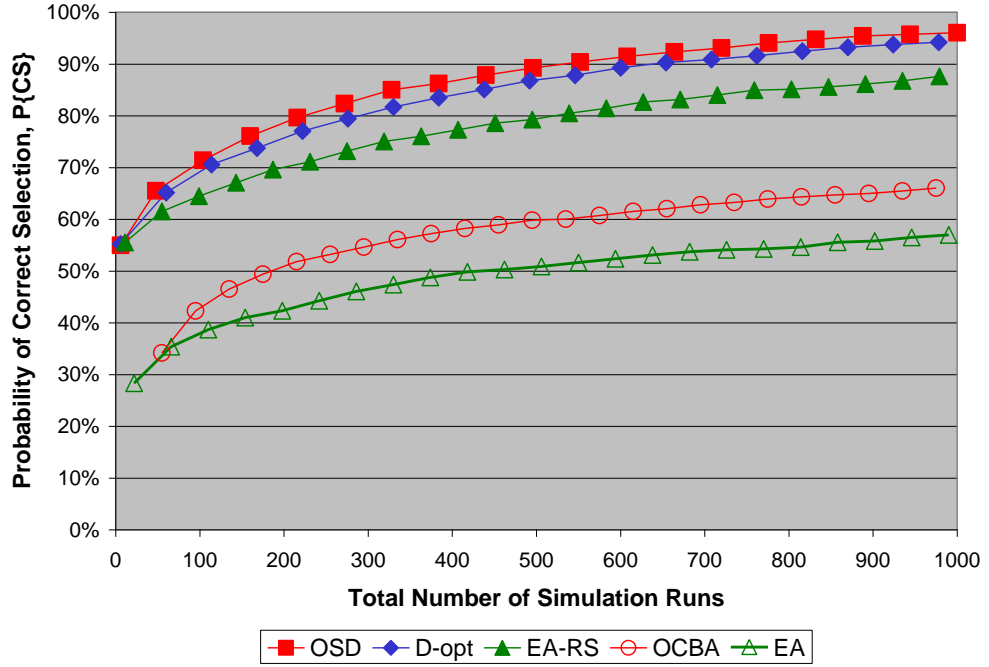
Just as in Experiment 1, the D-optimal method will choose  $x_1$ ,  $x_6$ , and  $x_{11}$  as the support points with allocation ratios  $\{\alpha_1, \alpha_6, \alpha_{11}\} = \{1/3, 1/3, 1/3\}$ . Since the support points are the same and the differences of the allocation rules between OSD and D-optimal are not very big, it is not surprising that the PCS results of these two rules are relatively close as compared with that in Experiment 1.

Table 2: OSD Allocations for Experiment 2

Total Runs	Design 1 ( $x = -1$ )	Design 6 ( $x = 0$ )	Design 11 ( $x = 1$ )
160	7.03%	49.99%	42.97%
440	7.00%	50.00%	43.00%
720	7.10%	50.00%	42.90%
1000	7.09%	50.00%	42.91%

### Experiment 3 ( $a = 1.0$ , 11 design locations)

Figure 7 presents the simulation results for Experiment 3. The allocation methods are the same as described for the previous two experiments and the results are similar. After 1,000 runs, D-optimal achieves a 94.3% PCS while OSD achieves the same PCS after only 804 runs.



**Figure 7: Results of Experiment 3**

For this experiment, the best design location will be  $x_{11} = 1.0$ , and so

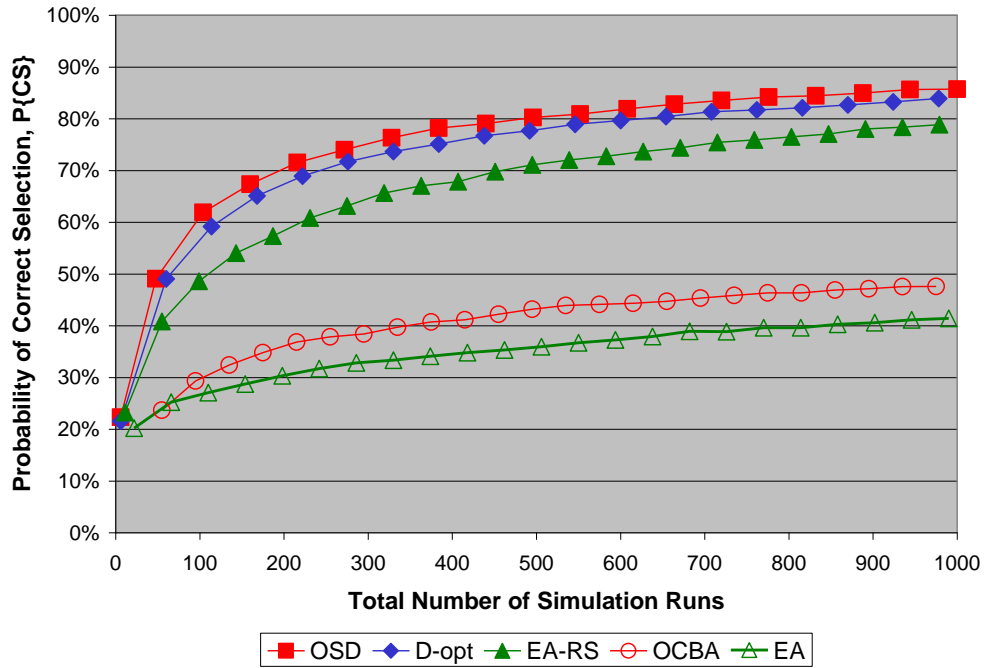
$x_A = x_1 = -1.0$  and  $x_Z = x_{10} = 0.8$ . Asymptotically, the OSD rule will again pick the same support points with D-optimal, i.e.,  $x_1$ ,  $x_6$ , and  $x_{11}$  but with

$\{\alpha_1, \alpha_6, \alpha_{11}\} = \{1/9, 1/2, 7/18\}$  instead of one third allocated to each support point. The support points for OSD and D-optimal are the same and the allocations for these two methods are again similar so the fact that the results are similar is consistent.

#### **Experiment 4 (randomly generated optimal solution, 11 design locations)**

In order to test the methods against a more diverse set of problems, the stationary point for the underlying quadratic equation is randomly selected for each of the 10,000

procedures from the distribution where  $a \sim U(-1,1)$ . The optimal solutions are the design locations closest to  $a$ . The allocation methods are the same as described for the previous experiments and the results shown in Figure 8 are consistent with the results from the first three experiments. After 1,000, runs D-optimal achieves an 84.2% PCS while OSD achieves the same PCS after only 776 runs.



**Figure 8: Results of Experiment 4**

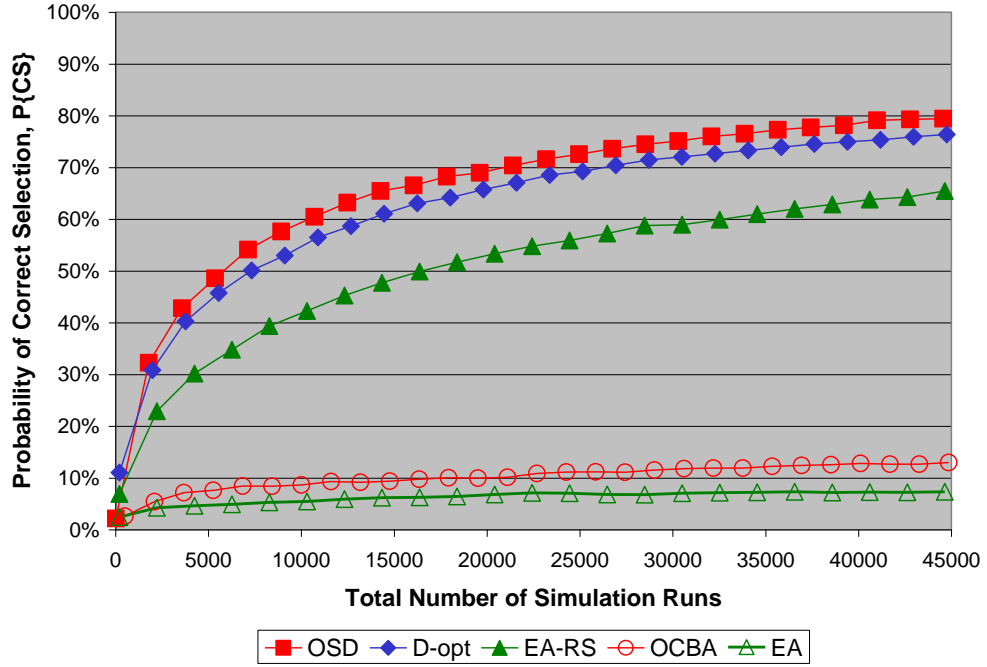
The results of this experiment represent a blend of the results from the three cases in (34). For the first two cases in (34), the support points and allocations for OSD will be very different than those for the D-optimality criterion and the performance of the OSD method will be comparatively better. For the “otherwise” case in (34), the support points

are the same for OSD and D-optimality. The allocations become more similar as the best design location gets closer to the boundary of the domain and consequently the results of the D-optimal method become more competitive with the OSD method.

#### **Experiment 5 (randomly generated optimal solution, 101 design locations)**

For this experiment, we used a domain of 101 design locations where  $x \in [-1, -0.98, \dots, 1]$  with the stationary point for the underlying quadratic equation again randomly selected for each of the 10,000 procedures from the distribution where  $a \sim U(-1, 1)$ . For OSD and OCBA, we provided initial runs as described above and then allocated 99 runs during additional iterations. The results are shown in Figure 9. With more designs across the same sized domain, the differences between the responses at the best design and its nearest competitors are smaller relative to the noise in the underlying function. Therefore, the PCS for each method is lower for this experiment than the experiments with only 11 design locations. However, the results are consistent with the other four experiments. After 45,000 runs, D-optimal achieves a 76.3% PCS while OSD achieves the same PCS after only 32,775 runs or about 73% of those required by D-optimal. This suggests that as OSD may perform relatively better as the number of design locations increases and is a potential area for future research.





**Figure 9: Results of Experiment 5**

### Experiment 6 (Non-quadratic Underlying Equation, 60 design locations)

For this last experiment, we partitioned the domain of 60 design locations into partitions with an equal number of design locations. We conducted the experiment using one, two, three, four, six, ten, twelve, and fifteen partitions. The partitions are disconnected such that, for example, the last design for the first partition when using six partitions is  $x_{10}$  and the first design location for the second partition is  $x_{11}$ . After constructing independent regression equations for each partition, we then allocate the simulation budget equally between the partitions and allocate using OSD within the partitions. Even with this inefficient allocation between the partitions, the results in Table 3 provide insight and demonstrate that OSD performs well if we can properly partition

the domain. Using partitioning schemes with one, four, six, ten, twelve, and fifteen partitions, the optimal design location is located in the middle half of its partition such that OSD will use the first two cases of (34) and construct a design symmetric about  $(x_M + x_b)/2$  as shown in Figure 4. The results in Table 3 suggest that OSD will do well if we can partition the domain so that the optimal design location is within the middle half of a partition and as close to the center of the partition as possible. To highlight this point, note the performance of OSD using twelve partitions and with the optimal design location in the center of the partition compared to the performance of OSD using six or ten partitions where the design location is not near the middle of the partition. Even with fewer runs allocated to the best partition for the twelve partition experiment, OSD provides a significantly higher PCS. Using partitioning schemes with two, three, and five partitions, the optimal design location is not in the middle half of the domain such that OSD uses the third case of (34) for the best partition. For these three schemes, OSD obtained a biased solution and a PCS of approximately zero.

Table 3: Results of Experiment 6

Number of Partitions	Optimal Design Location on the Partition	PCS after 2,000 runs
1	38 out of 60	100%
2	8 out of 30	0%
3	18 out of 20	0%
4	8 out of 15	98%
5	2 out of 12	0%
6	7 out of 10	18%
10	2 out of 6	5%
12	3 out of 5	34%
15	2 out of 4	13%

The next chapter will derive a means to efficiently allocate simulation runs on a partitioned domain instead of using an equal allocation scheme as used in this experiment. We will then revisit the impacts of the partitioning scheme on the performance of the derived allocation method for the partitioned domain with another numerical experiment.

## CHAPTER 4 OSD FOR PARTITIONED DOMAINS

Regression-based methods, including our OSD method developed in Chapter 3, are constrained with some typical assumptions such as an underlying quadratic function for the means and homogeneous simulation noise. For instances where the underlying function does not have a constant variance error, the least squares estimate will still be unbiased (Draper and Smith 1998). The OSD method will still asymptotically determine the best design location but may provide less than optimal allocations to do so. More problematic is if the underlying function is not quadratic to such an extent that we fail to find the best design location. However, to address a non-constant variance as well as a deviation from a quadratic nature, we can extend the OSD method by partitioning the design space into smaller domains in which the underlying function can be well approximated by a quadratic function and in which the variances are relatively the same. In a worst case scenario, the domain can be partitioned using sets of three design locations yielding an unbiased, piecewise quadratic estimate of the underlying function.

This chapter also explores a problem with the principal goal of selecting the best of multiple alternative design locations but on a partitioned domain. Without loss of generality, we assume that we have  $m$  adjacent partitions and that each partition has  $k$  design locations. We aim to find the minimization problem shown below in (35) where the “best” design location is the one with smallest expected performance measure

$$\min_{x_{hi}} y(x_{hi}) = E[f(x_{hi})]; x_{hi} \in [x_{11}, x_{12}, \dots, x_{1k}, x_{21}, x_{22}, \dots, x_{2k}, x_{m1}, x_{m2}, \dots, x_{mk}] . \quad (35)$$

Given the problem in (35), we aim to determine how to allocate between the partitions, how to allocate within the partitions, and which design locations to use as support points. Addressing how the domain is partitioned is not within the scope of this chapter and we assume this partitioning scheme is derived from knowledge of the domain, through iterative refinement such as a heuristic based upon the results of Experiment 6 in Chapter 3, or through an optimal selection procedure such as multivariate adaptive regression splines (MARS) (Friedman 1991).

#### 4.1. Problem Statement and Framework

We will use a notation and structure very similar to the one partition case described in Chapter 2 and Chapter 3. We consider that the expectation of the unknown underlying function for each partition is quadratic or approximately quadratic in nature on the prescribed domain, i.e., for each partition  $h$

$$y(x_{hi}) = \beta_{h0} + \beta_{h1}x_{hi} + \beta_{h2}x_{hi}^2. \quad (36)$$

For ease of notation, we define  $\beta_h = [\beta_{h0} \quad \beta_{h1} \quad \beta_{h2}]$ . In (36), the parameters  $\beta_h$  are unknown and we consider a common case where  $y(x_{hi})$  must be estimated via simulation with noise. The simulation output  $f(x_{hi})$  is independent from replication to replication such that

$$f(\mathbf{x}_{hi}) = \mathbf{y}(\mathbf{x}_{hi}) + \varepsilon_{hj}; \quad i = 1, \dots, k, \text{ where } \varepsilon_{hj} \sim N(0, \sigma_h^2). \quad (37)$$

The parameters  $\beta_h$  are unknown so  $y(x_{hi})$  are also unknown. However, we can find an estimated expected performance measure at  $x_{hi}$ , that we define as  $\hat{y}(x_{hi})$ , by using a least squares estimate of the form shown in (38) below where  $\hat{\beta}_{h0}$ ,  $\hat{\beta}_{h1}$ , and  $\hat{\beta}_{h2}$  are the least squares parameter estimates for the corresponding parameters associated with the constant, linear, and quadratic terms in (36).

$$\hat{y}(x_{hi}) = \hat{\beta}_{h0} + \hat{\beta}_{h1}x_{hi} + \hat{\beta}_{h2}x_{hi}^2. \quad (38)$$

In a similar manner, we define  $\hat{\beta}_h = [\hat{\beta}_{h0} \quad \hat{\beta}_{h1} \quad \hat{\beta}_{h2}]$ .

In order to obtain the least squares parameter estimates for each partition, we take  $n_h$  samples on any choice of  $x_{hi}$  (on at least three design locations for each partition to avoid singular solutions). We assume that these  $x_{hi}$  are given beforehand and we can only take samples from these points. Given the  $n_h$  samples, we define  $F_h$  as the  $n_h$  dimensional vector containing the replication output measures  $f(x_{hi})$  and  $X_h$  as the  $n_h \times 3$  matrix composed of rows consisting of  $[1 \quad x_{hi} \quad x_{hi}^2]$  with each row corresponding to its respective entry of  $f(x_{hi})$  in  $F_h$ . Using the matrix notation and a superscript  $t$  to indicate the transpose of a matrix, for each partition we determine the least squares estimate for the parameters  $\beta_h$  which minimize the sum of the squares of the error terms  $(F_h - X_h\beta_h)^t(F_h - X_h\beta_h)$ . We obtain the least squares estimate for the parameters as shown below:

$$\hat{\beta}_h = (X_h^t X_h)^{-1} X_h^t F_h.$$

Our problem is to select the design location associated with the smallest mean performance measure from among the  $mk$  design locations within the constraint of a computing budget with only  $T$  simulation replications. Given the least squares estimates for the parameters, we can use (38) to estimate the expected performance measure at each design location. We designate the design location with the smallest estimated performance measure in each partition as  $x_{hb}$  so that  $\hat{y}(x_{hb}) = \min_i \hat{y}(x_{hi})$  and designate  $x_{Bb}$  as the design location with the smallest estimated performance measure across the entire domain so that  $\hat{y}(x_{Bb}) = \min_h \hat{y}(x_{hb})$ . Given the uncertainty of the estimate of the underlying function,  $x_{Bb}$  is a random variable and we define Correct Selection as the event where  $x_{Bb}$  is indeed the best location. We define  $N_{hi}$  as the number of simulation replications conducted at design location  $x_{hi}$ . Since the simulation is expensive and the computing budget is restricted, we seek to develop an allocation rule for each  $N_{hi}$  that maximizes the *PCS* as is reflected in (39) below.

$$\begin{aligned}
& \max_{N_{11}, \dots, N_{mk}} \quad PCS = P\{ y(x_{Bb}) \leq y(x_{hi}) \forall h = 1, \dots, m, i = 1, \dots, k \} \\
& s.t. \quad \sum_{h=1}^m \sum_{i=1}^k N_{hi} = T
\end{aligned} \tag{39}$$

Elements of this problem are similar to the one partition case from the previous chapters. The constraint  $\sum_{h=1}^m \sum_{i=1}^k N_{hi} = T$  denotes the total computational cost and implicitly assumes that the simulation execution times for one sample are constant across the domain. Since the optimal allocation is dependent upon the uncertainty of the

parameters and the random variable  $x_{Bb}$ , we can only estimate the PCS even after exhausting the total simulation budget  $T$ .

In order to solve the problem in (39), we must obtain estimates for the parameters  $\beta_h$  and we will again use a Bayesian regression framework where the parameters  $\beta_h$  are assumed to be unknown and are treated as random variables. We aim to find the posterior distributions of  $\beta_h$  as the simulation replications are conducted and use these distributions to update the posterior distribution of the performance measures for each design location. We can then perform the comparisons with the performance measure at design location  $x_{Bb}$  as expressed in (39). We will use  $\tilde{\beta}_h$  and  $\tilde{y}(x_{hi})$  to denote the random variables whose probability distributions are the posterior distribution of  $\beta_h$  and  $y(x_{hi})$  conditional on  $F_h$  given samples respectively. Therefore, given a set of initial  $n_h$  simulation runs with the output contained in vector  $F_h$  and the design location  $x_{Bb}$  obtained from the least squares results, we can redefine the probability of correct selection from (39) based on the Bayesian concept (Chen et al. 2010 and He et al. 2007) as

$$PCS = P\{ \tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}) \forall h = 1, \dots, m, i = 1, \dots, k \} . \quad (40)$$

Using a non-informative prior distribution and assuming that the conditional distribution of the simulation output vector  $F_h$  is a multivariate normal distribution with mean  $X_h \beta_h$  and a covariance matrix  $\sigma_h^2 I$  where  $I$  is an identity matrix, the results from Chapter 2 show that the posterior distribution of  $\beta_h$  is then given by



$$\tilde{\beta}_h \sim N[(X_h^t X_h)^{-1} X_h^t F_h, \sigma_h^2 (X_h^t X_h)^{-1}]. \quad (41)$$

Since  $\tilde{y}(x_{hi})$  is a linear combination of the  $\tilde{\beta}_h$  elements, this means that  $\tilde{y}(x_{hi})$  has a Gaussian distribution of the form

$$\tilde{y}(x_{hi}) \sim N[X_{hi}(X_h^t X_h)^{-1} X_h^t F_h, \sigma_h^2 X_{hi}^t (X_h^t X_h)^{-1} X_{hi}] \quad (42)$$

where  $X_{hi}^t = [1 \quad x_{hi} \quad x_{hi}^2]$ .

Given the results of Theorem 2 in Chapter 3, we will refer to the support points for each partition as  $\{x_{h1}, x_{hs}, x_{hk}\}$  where  $x_{h1} < x_{hs} < x_{hk}$ . (Note that since  $x_{hb}$  may be at different locations on each partition, then  $x_{hs}$  may also be at different locations on each partition). For notation sake, we define the number of runs allocated to partition  $h$  as  $N_{h\bullet}$  and the percentage of  $N_{h\bullet}$  that is allocated to each support point as

$\alpha_{hi} = N_{hi} / N_{h\bullet}$ ,  $i \in \{1, s, k\}$ . Using this notation and the PCS equation in (40), we can now restate the OSD problem in (39) as the OSD problem in (43) below.

$$\begin{aligned} \mathbf{max} \quad & P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}) \forall h = 1, \dots, m, i = 1, \dots, k\} \\ \text{s.t.} \quad & \sum_{h=1}^m N_{h\bullet} (\alpha_{h1} + \alpha_{hs} + \alpha_{hk}) = T \end{aligned} \quad (43)$$

As with the one partition case, we can estimate  $\sigma_h^2$  from our least squares results and can calculate  $\tilde{y}(x_{hi})$  using (42). We can then use Monte Carlo simulation with (43) in order to estimate the PCS. The next section reduces the number of comparisons required and presents a way to approximate the PCS without running Monte Carlo Simulations.

## 4.2. A Lower Bound for PCS

Upon inspection, the PCS equation in (43) has two types of comparisons that are delineated in (44). The first type consists of the  $k - 1$  comparisons between  $\tilde{y}(x_{Bb})$  and each  $\tilde{y}(x_{Bi})$  for  $i \neq b$  in the best partition. The second type consists of the  $k(m - 1)$  comparisons between  $\tilde{y}(x_{Bb})$  and each  $\tilde{y}(x_{hi})$  when  $h \neq B$ .

$$PCS = P\{(\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{Bi}) \forall i \neq b) \cap (\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}) \forall h \neq B, i = 1, \dots, k)\} \quad (44)$$

Given  $x_{Bb}$  estimated from the second order polynomial metamodel results and Theorem 1 in Chapter 2, the assumption that our underlying function is quadratic within each partition allows us to reduce the required number of comparisons within the best partition from the  $k - 1$  comparisons expressed in (44) to 2 comparisons. As such, (44) can be rewritten as shown in (45) below subject to the three cases following (45).

$$PCS = P\{(\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{Bi}) \forall i = A, Z) \cap (\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}) \forall h \neq B, i = 1, \dots, k)\} \quad (45)$$

Case 1 (Interior Design Case):  $b \neq 1, k$ ;  $A = b - 1$ ;  $Z = b + 1$ ,

Case 2 (Left Boundary Design Case):  $b = 1$ ;  $A = 2$ ;  $Z = k$ , and

Case 3 (Right Boundary Design Case):  $b = k$ ;  $A = 1$ ;  $Z = k - 1$ .

We have the same assumption of an underlying function that is quadratic in the non-best partitions also. However, the comparisons in (44) for the non-best partitions are against  $\tilde{y}(x_{Bb})$  instead of the local best  $\tilde{y}(x_{hb})$ . If we apply the Bonferroni inequality to the comparisons with the global best for a non-best partition, we obtain

$$P\{ \tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k \} \geq 1 - \sum_{i=1}^k P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi}) \} \quad (46)$$

We can also establish a different lower bound for the comparisons from a non-best partition by using the quadratic information within the partition as expressed in the following lemma.

**Lemma 3:** Subject to the conditions expressed in Case 1 – Case 3 after (45), a lower bound for the comparisons with the global best for a non-best partition can be expressed by using the quadratic information within the partition as shown in (47).

$$P\{ \tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k \} \geq 1 - P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb}) \} - P\{ \tilde{y}(x_{hb}) \leq \tilde{y}(x_{hA}) \} - P\{ \tilde{y}(x_{hb}) \leq \tilde{y}(x_{hZ}) \} \quad (47)$$

Proof: We aim to prove a lower bound for the comparisons associated with a non-best partition that leverages the quadratic nature of the underlying function in the partition.

Subject to the conditions that follow (45), we will first show that

$$P\{ \tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k \} \geq P\{ (\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hb})) \cap (\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hA})) \cap (\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hZ})) \}$$

We begin by decomposing the comparisons into two parts: the comparison between the global best and the local best and the comparisons between the local best and the other designs.

$$P\{ \tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k \} = P\{ [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] + [\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i = 1, \dots, k \}$$

Highlighting the intersection with the global best and the local best, we obtain

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} = \\ P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0 \cap [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] + [\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i \neq b\}$$

Using the definition of conditional probability, we obtain

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} = P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \\ \cdot P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] + [\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i \neq b \mid [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \quad (48)$$

Given  $[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0$  as expressed in the conditional in (48), we know that

$$P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] + [\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i \neq b \mid [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \geq \\ P\{[\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i \neq b \mid [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\}$$

such that

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} \geq \\ P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \cdot P\{[\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i \neq b \mid [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \quad (49)$$

Using the results from Theorem 1 in Chapter 2, the  $k - 1$  comparisons in expressed in (49)

can be reduced to 2 comparisons subject to the same conditions expressed after (45).

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} \geq \\ P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \cdot P\{[\tilde{y}(x_{hb}) - \tilde{y}(x_{hi})] \leq 0, i = A, Z \mid [\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0\} \quad (50)$$

Using the definition of conditional probability, we can then rewrite (50) as an intersection of the three comparisons as shown in (51) or (52) below.

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} \geq \\ P\{[\tilde{y}(x_{Bb}) - \tilde{y}(x_{hb})] \leq 0 \cap [\tilde{y}(x_{hb}) - \tilde{y}(x_{hA})] \leq 0 \cap [\tilde{y}(x_{hb}) - \tilde{y}(x_{hZ})] \leq 0\} \quad (51)$$

or

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} \geq \\ P\{(\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hb})) \cap (\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hA})) \cap (\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hZ}))\} \quad (52)$$

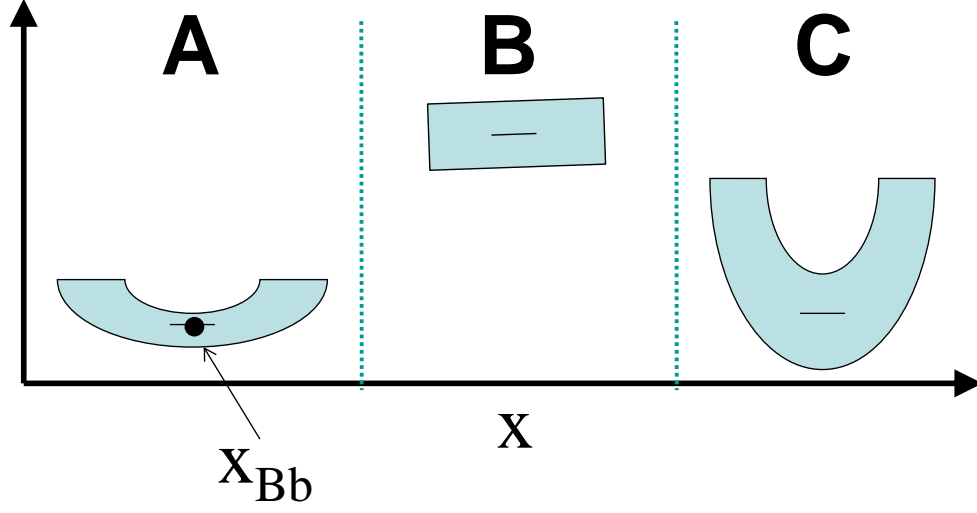
Applying the Bonferroni inequality to (52), we obtain

$$P\{\tilde{y}(x_{Bb}) \leq \tilde{y}(x_{hi}), i = 1, \dots, k\} \geq 1 - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb})\} - P\{\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hA})\} - P\{\tilde{y}(x_{hb}) \leq \tilde{y}(x_{hZ})\}.$$

For ease of discussion, we will refer to comparisons involving design locations from more than one partition as “between partition” comparisons and we will refer to comparisons involving design locations from just one partition as “within partition” comparisons.

In comparing the two different lower bounds expressed in (46) and (47), the lower bound in (47) allows us to reduce the number of comparisons from the  $k$  between partition comparisons in (46) to three comparisons: one between partition comparison and two within comparisons. The comparison between the global best and local best is common to both lower bound formulations. For each non-best partition, we would like to use the tightest lower bound in order to provide the best approximation of our PCS. Intuitively, when none of the between comparisons in (46) are competitive for a partition but the two within comparisons as expressed in (47) are very competitive, the formulation as expressed in (46) should provide a tighter lower bound. As an example of when we would intuitively prefer the bound provided by (46), consider Partition B in Figure 10 with a relatively flat underlying function within the partition but the performance measures for this partition are vastly removed from the performance measure of the global best design location in Partition A. At the other extreme, consider comparisons between Partition C and Partition A in Figure 10. The formulation in (47) intuitively should provide a tighter lower bound when many of the between comparisons in (46) are

competitive but the two within comparisons in (47) are not. We will explore the conditions of when to use each type of formulation more in Section 4.4 as we discuss implementation of the derived method.



**Figure 10: Intuitive Lower Bound Scenarios**

Given these two possible lower bounds for each non-best partition, applying the Bonferroni inequality to (45) yields a lower bound for our PCS as shown in (53), which we will consider our approximate PCS (APCS).

$$APCS = 1 - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA})\} - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ})\} - \sum_{h \neq B} \min \left( \begin{array}{c} \sum_{i=1}^k P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi})\}, \\ P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb})\} + P\{\tilde{y}(x_{hb}) \geq \tilde{y}(x_{hA})\} + P\{\tilde{y}(x_{hb}) \geq \tilde{y}(x_{hZ})\} \end{array} \right) \quad (53)$$

To simplify the notation later in the paper, we will define the set of partitions  $\psi$  as those partitions where we use the lower bound associated with (47) such that

$$\psi = \left\{ h : \left( \sum_{i=1}^k P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi}) \} \right) \geq \left( P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hA}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hZ}) \} \right) \right\}$$

Using this definition of  $\psi$ , we can write (53) in an alternate form as

$$\begin{aligned} APCS = & 1 - P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA}) \} - P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ}) \} - \sum_{h \neq B, h \notin \psi} \sum_{i=1}^k P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi}) \} \\ & - \sum_{h \neq B, h \in \psi} P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hA}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hZ}) \} \end{aligned}$$

Using this alternate form of the APCS in (53), we can now restate the POSD problem in (45) as the POSD problem in (54) below.

#### POSD Problem

$$\begin{aligned} \max_{N_{h1}, N_{hs}, N_{hk} \forall h=1, \dots, m} \quad & APCS = 1 - P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA}) \} - P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ}) \} \\ & - \sum_{h \neq B, h \notin \psi} \sum_{i=1}^k P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi}) \} \\ & - \sum_{h \neq B, h \in \psi} P\{ \tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hb}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hA}) \} + P\{ \tilde{y}(x_{hb}) \geq \tilde{y}(x_{hZ}) \} \\ \text{s.t.} \quad & \sum_{h=1}^m N_{h\bullet} (\alpha_{h1} + \alpha_{hs} + \alpha_{hk}) = T; \\ & \alpha_{hi} = 0 \forall i \neq 1, s, k \end{aligned}$$

with (54)

Case 1 (Interior Design Case):  $b \neq 1, k$ ;  $A = b - 1$ ;  $Z = b + 1$ ,

Case 2 (Left Boundary Design Case):  $b = 1$ ;  $A = 2$ ;  $Z = k$ , and

Case 3 (Right Boundary Design Case):  $b = k$ ;  $A = 1$ ;  $Z = k - 1$ .

### 4.3. Lagrangian Formulation for APCS

Since our aim is to efficiently allocate the computing budget to the three support points in each partition,  $\{x_{h1}, x_{hs}, x_{hk}\}$ , we will rewrite the APCS equation in (54) so that it is expressed in terms of the number of simulation runs allocated to each partition and the percentage of these partition allocations that is allocated to each support point within the partitions.

For the within partition comparisons, define

$$\tilde{d}(x_{hi}) \equiv \tilde{y}(x_{hi}) - \tilde{y}(x_{hb}) = \tilde{\beta}_{h2}(x_{hi}^2 - x_{hb}^2) + \tilde{\beta}_{h1}(x_{hi} - x_{hb}).$$

Using the results of Section 3.3,

$$P\{-\tilde{d}(x_{hi}) \geq 0\} = \int_{\frac{\hat{d}(x_{hi})}{\sqrt{\varsigma_{hi}}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{r^2}{2}} dr. \quad (55)$$

and

$$\varsigma_{hi} = \frac{\sigma_h^2}{N_{h\bullet}} \left[ \frac{D_{hi,1}^2}{\alpha_{h1}} + \frac{D_{hi,s}^2}{\alpha_{hs}} + \frac{D_{hi,k}^2}{\alpha_{hk}} \right],$$

where

$$\begin{aligned} D_{hi,1} &= \left\{ \frac{(x_{hs} - x_{hi})(x_{hk} - x_{hi}) - (x_{hs} - x_{hb})(x_{hk} - x_{hb})}{(x_{h1} - x_{hs})(x_{h1} - x_{hk})} \right\}, \\ D_{hi,s} &= \left\{ \frac{(x_{h1} - x_{hi})(x_{hk} - x_{hi}) - (x_{h1} - x_{hb})(x_{hk} - x_{hb})}{(x_{hs} - x_{h1})(x_{hs} - x_{hk})} \right\}, \\ D_{hi,k} &= \left\{ \frac{(x_{h1} - x_{hi})(x_{hs} - x_{hi}) - (x_{h1} - x_{hb})(x_{hs} - x_{hb})}{(x_{hk} - x_{h1})(x_{hk} - x_{hs})} \right\}. \end{aligned} \quad (56)$$



For the between partition comparisons, define  $\tilde{\delta}(x_{hb}) \equiv \tilde{y}(x_{hb}) - \tilde{y}(x_{Bb})$ . As with the within partition comparisons, this result shows that  $\tilde{\delta}(x_{hb})$  is a linear combination of the  $\tilde{\beta}_h$  elements so the  $\tilde{\delta}(x_{hb})$  terms are also normally distributed. Using the results of Chapter 2,  $\hat{\delta}(x_{hb}) \sim N[\hat{\delta}(x_{hb}), \xi_{hb}]$  where  $\hat{\delta}(x_{hb}) \equiv \hat{y}(x_{hb}) - \hat{y}(x_{Bb})$ . Assuming independence of the simulation runs between partitions,

$$\xi_{hb} = \sigma_h^2 \begin{pmatrix} 1, & x_{hb}, & x_{hb}^2 \end{pmatrix} (X_h^t X_h)^{-1} \begin{pmatrix} 1 \\ x_{hb} \\ x_{hb}^2 \end{pmatrix} + \sigma_B^2 \begin{pmatrix} 1, & x_{Bb}, & x_{Bb}^2 \end{pmatrix} (X_B^t X_B)^{-1} \begin{pmatrix} 1 \\ x_{Bb} \\ x_{Bb}^2 \end{pmatrix}. \quad (57)$$

Similar to the within partition comparison,

$$P\{-\tilde{\delta}(x_{hb}) \geq 0\} = \int_{\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt. \quad (58)$$

As we take additional simulation runs,  $E[\tilde{\delta}(x_{hb})] = \hat{\delta}(x_{hb})$  and the information obtained by additional simulation runs will affect  $\xi_{hb}$  and thus our *PCS* criterion. Likewise, we may write (57) in the simplified form:

$$\xi_{hb} = \frac{\sigma_h^2}{N_{h\bullet}} \left[ \frac{E_{hb,1}^2}{\alpha_{h1}} + \frac{E_{hb,s}^2}{\alpha_{hs}} + \frac{E_{hb,k}^2}{\alpha_{hk}} \right] + \frac{\sigma_B^2}{N_{B\bullet}} \left[ \frac{E_{Bb,1}^2}{\alpha_{B1}} + \frac{E_{Bb,s}^2}{\alpha_{Bs}} + \frac{E_{Bb,k}^2}{\alpha_{Bk}} \right],$$

where  $E_{hi,1} = \left\{ \frac{(x_{hs} - x_{hi})(x_{hk} - x_{hi})}{(x_{h1} - x_{hs})(x_{h1} - x_{hk})} \right\}, \quad E_{hi,s} = \left\{ \frac{(x_{h1} - x_{hi})(x_{hk} - x_{hi})}{(x_{hs} - x_{h1})(x_{hs} - x_{hk})} \right\},$

$$E_{hi,k} = \left\{ \frac{(x_{h1} - x_{hi})(x_{hs} - x_{hi})}{(x_{hk} - x_{h1})(x_{hk} - x_{hs})} \right\}. \quad (59)$$

By expressing both the within partition comparisons and the between partition comparisons in terms of the number of simulation runs allocated to each partition and the percentages of the simulation budget allocated to each support point, we can define a Lagrangian function

$$Q = APCS + \lambda \left[ T - \sum_{h=1}^m N_{h\bullet} (\alpha_{h1} + \alpha_{hs} + \alpha_{hk}) \right]$$

or, using (55) and (58), as

$$Q = 1 - \left[ \int_{\frac{\hat{d}(x_{BA})}{\sqrt{\varsigma_{BA}}}}^{\infty} \phi(r) dr + \int_{\frac{\hat{d}(x_{BZ})}{\sqrt{\varsigma_{BZ}}}}^{\infty} \phi(r) dr \right] - \sum_{h \neq B, h \notin \psi} \sum_{i=1}^k \int_{\frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}}}^{\infty} \phi(r) dr - \sum_{h \neq B, h \in \psi} \int_{\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}}^{\infty} \phi(r) dr + \int_{\frac{\hat{d}(x_{hA})}{\sqrt{\varsigma_{hA}}}}^{\infty} \phi(r) dr + \int_{\frac{\hat{d}(x_{hZ})}{\sqrt{\varsigma_{hZ}}}}^{\infty} \phi(r) dr + \lambda \left[ T - \sum_{h=1}^m N_{h\bullet} (\alpha_{h1} + \alpha_{hs} + \alpha_{hk}) \right] \quad (60)$$

We can investigate  $\frac{\partial Q}{\partial \alpha_{hj}}$  to determine the within partition allocations and then  $\frac{\partial Q}{\partial N_{h\bullet}}$  to

determine the between partition allocations. For example, by setting  $\frac{\partial Q}{\partial \alpha_{hj}} = 0$  and

$\frac{\partial Q}{\partial N_{h\bullet}} = 0$  for when  $h \neq B$  and  $h \in \psi$ , we obtain

$$\begin{aligned} & \phi\left(\frac{\hat{d}(x_{hA})}{\sqrt{\varsigma_{hA}}}\right) \frac{\hat{d}(x_{hA})}{(\varsigma_{hA})^{3/2}} \frac{D_{hA,j}^2}{\alpha_{hj}^2} + \phi\left(\frac{\hat{d}(x_{hZ})}{\sqrt{\varsigma_{hZ}}}\right) \frac{\hat{d}(x_{hZ})}{(\varsigma_{hZ})^{3/2}} \frac{D_{hZ,j}^2}{\alpha_{hj}^2} \\ & + \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{(\xi_{hb})^{3/2}} \frac{E_{hb,j}^2}{\alpha_{hj}^2} = \frac{2\lambda (N_{h\bullet})^2}{\sigma_h^2} \end{aligned} \quad (61)$$

and

$$\begin{aligned}
& \phi\left(\frac{\hat{d}(x_{hA})}{\sqrt{\varsigma_{hA}}}\right) \frac{\hat{d}(x_{hA})}{(\varsigma_{hA})^{3/2}} \left[ \frac{D_{hA,l}^2}{\alpha_{hl}} + \frac{D_{hA,s}^2}{\alpha_{hs}} + \frac{D_{hA,k}^2}{\alpha_{hk}} \right] \\
& + \phi\left(\frac{\hat{d}(x_{hZ})}{\sqrt{\varsigma_{hZ}}}\right) \frac{\hat{d}(x_{hZ})}{(\varsigma_{hZ})^{3/2}} \left[ \frac{D_{hZ,l}^2}{\alpha_{hl}} + \frac{D_{hZ,s}^2}{\alpha_{hs}} + \frac{D_{hZ,k}^2}{\alpha_{hk}} \right] \\
& + \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{(\xi_{hb})^{3/2}} \left[ \frac{E_{hb,l}^2}{\alpha_{hl}} + \frac{E_{hb,s}^2}{\alpha_{hs}} + \frac{E_{hb,k}^2}{\alpha_{hk}} \right] = \frac{2\lambda(N_{h\bullet})^2}{\sigma_h^2}
\end{aligned} \tag{62}$$

However, the equations obtained in (61) and (62) are nonlinear with respect to  $\alpha_{hj}$  and

$N_{h\bullet}$ . Our aim is to provide a simple means for efficiently allocating the budget instead of

attempting to use a root finding technique to numerically compute the optimal values for

$\alpha_{hj}$  and  $N_{h\bullet}$ . In the next section, we will derive approximations for our optimal

allocations and our numerical results in Section 4.6 will demonstrate their efficiency.

#### 4.4. Approximately Optimal Allocations

The within partition allocations intuitively have three purposes to address the problem as expressed in (54) or (60):

- to determine the best design location within the partition,
- to provide the most accurate estimate of the performance measure at the best design location within the partition for use in the between partition comparisons,
- and to minimize the probability of the worst comparison in the partition.

While these three purposes are related, their associated objective functions are different. The results in Chapter 3 demonstrate that the OSD conditions in (33) and (34) provide approximately optimal allocations to find the best design location in the partition. On the other hand, as shown in Lemma 4 and Lemma 5 in Appendix D, providing all of the runs to the best local design or design with the worst comparison in a partition provide approximately optimal allocations to satisfy the second and third criteria. These generalizations are captured more formally in Theorem 6 below.

In this section, we will derive an efficient heuristic based upon approximately optimal allocations of simulation runs to the designated support points  $\{x_{h1}, x_{hs}, x_{hk}\}$  by analyzing allocations of lower and upper bounds of the *APCS*. We denote  $L$  and  $U$  as the lower and upper bounds respectively.

To simplify the notation, we define probability  $P_{\Omega} = \max_{h \neq B; i=1, \dots, k} [P\{-\tilde{\delta}(x_{hi}) \geq 0\}]$ . This probability is the most competitive comparison from among the  $(m-1)k$  between partition comparisons.

We also define for the best partition probability  $P_{BM}$  and, for  $h \neq B$ , we define probability  $P_{hM}$  such that for  $h = B$

$$M = \arg \max_{A, Z} [P\{-\tilde{d}(x_{BA}) \geq 0\}, P\{-\tilde{d}(x_{BZ}) \geq 0\}]$$

$$P_{BM} = \max [P\{-\tilde{d}(x_{BA}) \geq 0\}, P\{-\tilde{d}(x_{BZ}) \geq 0\}] \quad (63a)$$

and, for  $h \neq B$

$$M = \begin{cases} \mathbf{arg\,max}_{A,Z,b} [P\{-\tilde{d}(x_{hA}) \geq 0\}, P\{-\tilde{d}(x_{hZ}) \geq 0\}, P\{-\tilde{\delta}(x_{hb}) \geq 0\}], & h \in \psi \\ \mathbf{arg\,max}_{i=1,\dots,k} [P\{-\tilde{\delta}(x_{hi}) \geq 0\}], & \text{otherwise} \end{cases}$$

$$P_{hM} = \begin{cases} \mathbf{max} [P\{-\tilde{d}(x_{hA}) \geq 0\}, P\{-\tilde{d}(x_{hZ}) \geq 0\}, P\{-\tilde{\delta}(x_{hb}) \geq 0\}], & h \in \psi \\ \mathbf{max} [P\{-\tilde{\delta}(x_{hi}) \geq 0\}], & \text{otherwise} \end{cases} \quad (63b)$$

Finally, we define  $R_B$  as the associated signal to noise ratio for the best partition and  $R_h$  as the associated signal to noise ratio for  $h \neq B$  as shown below.

$$R_B = \frac{(\hat{d}(x_{BM}))^2}{\sigma_B^2 [ |D_{BM,l}| + |D_{BM,s}| + |D_{BM,k}| ]^2} \quad (64a)$$

$$R_h = \begin{cases} \frac{(\hat{d}(x_{hM}))^2}{\sigma_h^2 [ |D_{hM,l}| + |D_{hM,s}| + |D_{hM,k}| ]^2}, & \begin{matrix} P_{hM} = P\{-\tilde{d}(x_{hA}) \geq 0\}, \\ P_{hM} = P\{-\tilde{d}(x_{hZ}) \geq 0\} \end{matrix} \\ \frac{(\hat{\delta}(x_{hb}))^2}{\sigma_h^2}, & \text{otherwise} \end{cases} \quad (64b)$$

**Theorem 6:** Using lower and upper bounds of the APCS, approximately optimal between partition allocations and within partition allocations are as shown in (65a) and (65b) for the best partition and (66a) and (66b) for  $h \neq B$ . For brevity, we use OSD to refer to allocations in accordance with the OSD conditions in (33) and (34) as derived for the one partition case in Chapter 3.

For  $h = B$ ,

$$N_{B\bullet} = \begin{cases} T, & P_{\text{BM}} \geq P_{\Omega} \\ \frac{R_j}{R_i} N_{j\bullet}, & P_{\text{BM}} < P_{\Omega}; h \in \psi, M = A, Z \forall h \\ \sigma_B^2 \sqrt{\sum_{h \notin \psi}^m \frac{(N_{h\bullet})^2}{\sigma_h^2} + \sum_{h \in \psi, M=b}^m \frac{(N_{h\bullet})^2}{\sigma_h^2}}, & \text{otherwise} \end{cases} \quad (65a)$$

$$\alpha_{Bi} = \begin{cases} OSD, & P_{\text{BM}} \geq P_{\Omega} \\ OSD, & P_{\text{BM}} < P_{\Omega}; h \in \psi, M = A, Z \forall h \\ \alpha_{Bb} = 1.0, & \text{otherwise} \end{cases} \quad (65b)$$

For  $h \neq B$ ,

$$N_{h\bullet} = \begin{cases} 0 & P_{\text{BM}} \geq P_{\Omega} \\ \frac{R_j}{R_h} N_{j\bullet} & \text{otherwise} \end{cases} \quad (66a)$$

$$\alpha_{hi} = \begin{cases} OSD, & P_{hM} = P\{-\tilde{d}(x_{hA}) \geq 0\}, P_{hM} = P\{-\tilde{d}(x_{hZ}) \geq 0\} \\ \alpha_{hM} = 1.0, & \text{otherwise} \end{cases} \quad (66b)$$

Proof: We will examine the three cases that are delineated in (65a). The first case in (65a) is a general case where  $P_{\text{BM}} \geq P_{\Omega}$ . The second case is a special case that addresses where  $P_{\text{BM}} < P_{\Omega}$  but there are no between partition comparisons in the lower and upper bounds of the APCS (such that  $h \in \psi \forall h$  and  $M = A, Z \forall h$ ). The third case is a general case where  $P_{\text{BM}} < P_{\Omega}$ , except for the special case addressed by the second case. As such, our proof will establish the first and third cases and then address the special case.

Case 1:  $P_{\text{BM}} \geq P_{\Omega}$

When  $P_{\text{BM}} \geq P_{\Omega}$ , we will not use the quadratic bound formulation for any of the non-best comparisons (such that  $h \notin \psi$  for every  $h \neq B$ ). Therefore, our APCS from (14) simplifies to

$$APCS \geq 1 - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA})\} - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ})\} - \sum_{h \notin \psi} \sum_{i=1}^k P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi})\}.$$

To establish the upper bound, we show that

$$\begin{aligned} & 1 - \max[P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA})\}, P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ})\}] \geq \\ & 1 - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA})\} - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ})\} - \sum_{h \notin \psi} \sum_{i=1}^k P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi})\} \end{aligned}$$

For the lower bound, given  $P_{\text{BM}} \geq P\{-\tilde{\delta}(x_{hi}) \geq 0\} \forall h \neq B, i = 1, \dots, k$ , we know that

$$\begin{aligned} & 1 - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BA})\} - P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{BZ})\} - \sum_{h \notin \psi} \sum_{i=1}^k P\{\tilde{y}(x_{Bb}) \geq \tilde{y}(x_{hi})\} \geq \\ & 1 - P_{\text{BM}} - P_{\text{BM}} - \sum_{h \notin \psi} \sum_{i=1}^k P_{\text{BM}} \end{aligned}$$

Therefore, when  $P_{\text{BM}} \geq P_{\Omega}$ , we can use the lower and upper bounds shown in (67).

$$1 - \{(m-1)k + 2\}P_{\text{BM}} \leq APCS \leq 1 - P_{\text{BM}} \quad (67)$$

Since  $L$  and  $U$  only contain within comparisons for the best partition, we obtain

$N_{B\bullet} = T$  and  $N_{h\bullet} = 0$  for  $h \neq B$ . For the proof of the within allocation of  $\alpha_{Bi}$  in accordance with the OSD conditions, see Chapter 3.

Case 3:  $P_{\text{BM}} < P_{\Omega}$  (except for the special case addressed by Case 2 below)

When  $P_{\text{BM}} < P_{\Omega}$ , the lower and upper bounds shown in (68) can be established using a very similar approach as used for when  $P_{\text{BM}} \geq P_{\Omega}$ .

$$1 - 2P_{\Omega\text{M}} - 3 \sum_{h \in \psi} P_{h\text{M}} - k \sum_{h \notin \psi} P_{h\text{M}} \leq \text{APCS} \leq 1 - \sum_{h \in \psi} P_{h\text{M}} - \sum_{h \notin \psi} P_{h\text{M}} \quad (68)$$

For the within partition allocations, see the results from Chapter 3 for when  $P_{h\text{M}} = P\{-\tilde{d}(x_{h\text{A}}) \geq 0\}$  or when  $P_{h\text{M}} = P\{-\tilde{d}(x_{h\text{Z}}) \geq 0\}$ . Lemma 4 in Appendix D addresses when  $P_{h\text{M}} = P\{-\tilde{\delta}(x_{h\text{i}}) \geq 0\}$  which occurs when  $h \notin \psi$  or when  $h \in \psi$  and  $P_{h\text{M}} = P\{-\tilde{\delta}(x_{h\text{b}}) \geq 0\}$ . For the allocations within the best partition, except for the special case, see Lemma 5 in Appendix D. The proof for the between partition allocations closely follows those presented in Chen et al (2000) and Glynn and Juneja (2004). See Lemma 6 and Lemma 7 in Appendix E.

Case 2:  $P_{\text{BM}} < P_{\Omega}$ ,  $h \in \psi \forall h$  and  $\text{M} = \text{A}, \text{Z} \forall h$

For the special case when  $P_{\text{BM}} < P_{\Omega}$  but there are no between partition comparisons in the APCS (such that  $h \in \psi \forall h$  and  $\text{M} = \text{A}, \text{Z} \forall h$ ), we also use the lower and upper bounds presented in (68). However, the allocations for the best partition are obtained using the same approach presented in Lemma 6 in Appendix E for the non-best partitions.

The results for the between partition allocations presented in (65a) and (66a) for  $P_{\text{BM}} < P_{\Omega}$  are very similar to those obtained by Chen et al (2000) and Glynn and Juneja



(2004) for traditional OCBA. The rule for the best partition in (65a) has a “square root” rule similar to the one obtained by the traditional OCBA method. The key difference is that terms are not included for partitions associated when  $P_{hM} = P\{-\tilde{d}(x_{hA}) \geq 0\}$  or when  $P_{hM} = P\{-\tilde{d}(x_{hZ}) \geq 0\}$ . The comparisons associated with these partitions do not involve the global best. The allocations for the non-best partitions in (66a) also have a “ratio” rule that is similar to the one obtained by the traditional OCBA method. However, this ratio rule uses a difference in the expected performance measure derived from the regression equation instead of the mean of the samples at the design location. The variance terms are also derived from the regression equation and influenced by the allocation methods through the Lagrange polynomial coefficients.

Given the results from Theorem 6 expressed in (67a-b) for the non-best partitions, we now revisit how to approximate the probabilities in (46) and (47) in order to choose between the two different lower bounds expressed in the APCS in (54) or (60). As an approximation, we use the Cantelli inequality (Spall 2003) such that for within partition comparisons

$$P\{-\tilde{d}(x_{hi}) \geq 0\} \leq \frac{1}{1 + \left( \frac{\hat{d}(x_{hi})}{\sqrt{\varsigma_{hi}}} \right)^2}.$$

A similar expression can be made for between partition comparisons such that for each partition  $h$  in (54) or (60), we seek

$$\min \left( \sum_{i=1}^k \frac{1}{1 + \left( \frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}} \right)^2}, \frac{1}{1 + \left( \frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}} \right)^2} + \frac{1}{1 + \left( \frac{\hat{d}(x_{hA})}{\sqrt{\xi_{hA}}} \right)^2} + \frac{1}{1 + \left( \frac{\hat{d}(x_{hZ})}{\sqrt{\xi_{hZ}}} \right)^2} \right). \quad (69)$$

While (69) is based upon the APCS formulation in Section 4.3, for an alternate approach we can utilize the results of Section 4.4 and base our choice on only  $P_{hM}$ . Therefore, for each non-best partition  $h$  in (54) or (60), we seek

$$\min \left( \max_{i=1 \dots k} \frac{1}{1 + \left( \frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}} \right)^2}, \max \left[ \frac{1}{1 + \left( \frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}} \right)^2}, \frac{1}{1 + \left( \frac{\hat{d}(x_{hA})}{\sqrt{\xi_{hA}}} \right)^2}, \frac{1}{1 + \left( \frac{\hat{d}(x_{hZ})}{\sqrt{\xi_{hZ}}} \right)^2} \right] \right). \quad (70)$$

For the numerical experiments presented in Section 4.6, we experimented with both of the approximations presented in (69) and (70) to determine  $h \in \psi$  and obtained nearly identical results.

#### 4.5. POSD Procedure

The following is the algorithm that we used to implement the POSD method for the experiments in this dissertation. The actual ProModel code used is in Appendix G.

### **POSD Procedure (Maximizing PCS)**

**INPUT**             $k$  (the number of design locations),  $T$  (the computing budget),  $x_{hi}$  (the design locations with partitions already determined),  $n_0$  (the number of initial runs),  $\theta_j$  (the number runs allocated each iteration  $j$ );

**INITIALIZE**     $j \leftarrow 0$ ;

Perform  $n_0$  simulation replications for three design locations in each partition; by convention we use the D-opt support points such that

$$\alpha_{h1}^j = \alpha_{h \frac{(k+1)}{2}}^j = \alpha_{hk}^j = n_0 / 3.$$

**LOOP**            **WHILE**  $\sum_{i=1}^k N_i^j < T$  **DO**

**UPDATE**    - Estimate a quadratic regression equation using the information from all prior simulation runs for each partition.

    - Estimate the mean and variance of each design location using

$$\hat{y}(x_{hi}) = \hat{\beta}_{h0} + \hat{\beta}_{h1}x_i + \hat{\beta}_{h2}x_i^2.$$

    - Determine the observed global best design so that  $x_{Bb} = \mathbf{argmin}_{Bi} \hat{y}(x_i)$

    and the local best design in each partition so that  $x_{hb} = \mathbf{argmin}_{hi} \hat{y}(x_i)$ .

    - Based upon the location of the best design in each partition, use (54) to determine  $x_{hA}$  and  $x_{hZ}$ .

    - Determine  $P_{BM}$  and  $P_{hM}$  using (64a-b) and corresponding  $R_B$  and  $R_h$  using (65a-b).

- Determine  $P_{\Omega} = \arg \max_{h \neq B; i=1 \dots k} [P\{-\tilde{\delta}(x_{hi}) \geq 0\}]$ .

- Determine  $h \in \psi$  using (70).

**ALLOCATE** Increase the computing budget by  $\theta_{j+1}$  and calculate the new between

budget allocations  $N_{h\bullet}^{j+1}$  using (65a) and (66a) (round as needed).

Using  $N_{h\bullet}^{j+1}$  as well as (65b) and (66b), determine the within budget

allocations for  $\alpha_{h1}^{j+1}$ ,  $\alpha_{hs}^{j+1}$  and  $\alpha_{hk}^{j+1}$  (round as needed).

**SIMULATE** Perform  $\alpha_{hi}^{j+1}$  simulations for partition  $h$ ,  $h=1, \dots, m$ ; design  $i$ ,  $i = 1, s$ ,

$k$ ;  $j \leftarrow j+1$ .

**END OF LOOP**

#### 4.6. Numerical Testing and Results

In this section, we describe how we compared the results from our new POSD method against the results from five other allocation procedures. We start by providing a description of the other methods chosen to provide a perspective of the efficiency gained by our POSD method. We then describe our testing framework and provide our experimental results.

The first two other allocation procedures are the equal allocation method (EA) and the OCBA method. These methods were described in Section 3.7 and require no modification for use with the partitioned domain. Both EA and OCBA rely upon comparisons of the mean response at the global best design location and each individual

design location and do not rely upon a response surface within each partition to aid in the comparisons.

For our experiments, we will also compare against three methods that utilize a response surface within partitions: equal allocation with a response surface (EA-RS), D-opt, and OSD. The first two methods were described in Section 3.7 and OSD is the method derived in Chapter 3. The three methods were adapted for the partitioned case by utilizing the respective methods within the partitions but equally allocating between the partitions.

For the POSD method, we initialize as per the algorithm described in the previous section with  $N_{h1} = N_{h,(k+1)/2} = N_{hk} = n_0 = 20$ . We then used the algorithm described in Section 4.5 to allocate an additional 84 runs between each partition and within each partition.

The first experiment considers a function with three local minima on a domain with 60 design locations and compares the results of using POSD against the other five methods described above. The second experiment uses the same domain and underlying function as the first experiment but the simulation noises are not normally distributed. The third experiment also uses the same domain and underlying function as the first experiment but analyzes the impact of our partitioning scheme similar to Experiment 6 in Chapter 3. The fourth experiment also compares the results of using POSD against the other five methods described above. This experiment, also with 60 design locations, has only one global minimum but the function is not relatively symmetric about the optimum location except in a local area. For the first, second, and fourth experiments, we used a

heuristic based upon the results of Experiment 6 in Chapter 3 that exploits the adaptive nature of the first two cases of (34) and partitioned the domains of 60 design locations of the experiments into six disconnected partitions such that, for example, the last design location for the first partition is  $x_{10}$  and the first design location for the second partition is  $x_{11}$ .

We conducted all four experiments using a total computing budget of 10,000 runs. The results will show that these amounts are sufficient to compare the performance of the methods and determine the sensitivity of the POSD to the assumption of normally distributed noises. To mitigate the fact that the different allocation methods have varying fixed costs associated with them and in order to compare the performance of the methods using various simulation budgets, we calculate the PCS for each method during each iteration until the total simulation budget is exhausted. We repeat this whole procedure 10,000 times and then estimate the PCS obtained for each method after these 10,000 independent applications.

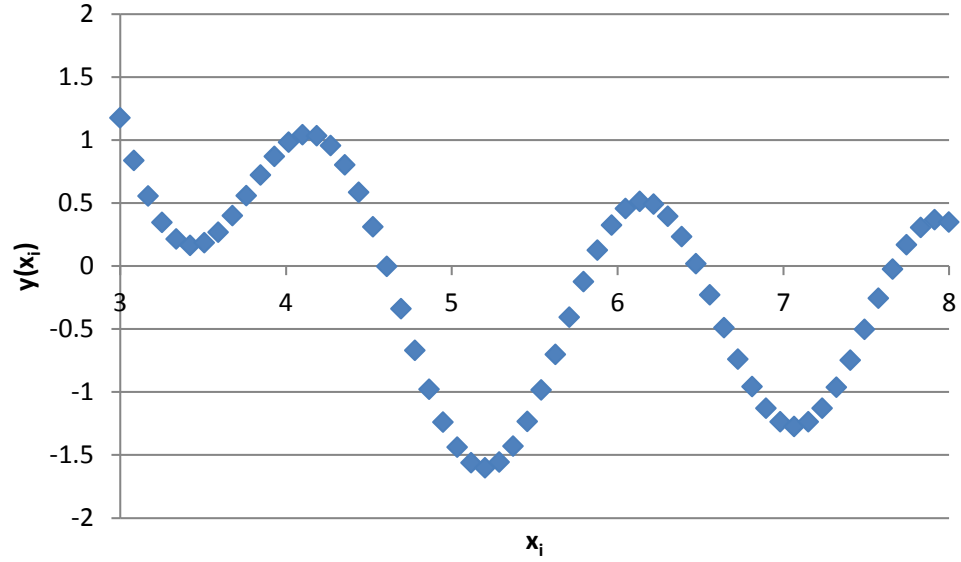
#### **Experiment 7 (three local minima, 60 design locations)**

This experiment is taken from the global optimization literature (Törn and Žilinskas, 1989) and uses the following function:

$$f(x_i) = \sin(x_i) + \sin(10x_i / 3) + \ln(x_i) - 0.84x_i + 3 + N(0,1) .$$

We used a domain consisting of 60 evenly spaced design locations where  $\mathbf{x} \in [3, 8]$  such that the global minimum is  $x_{27} \approx 5.20$  and  $y(x_{27}) \approx -1.60$ . Figure 11 below shows that

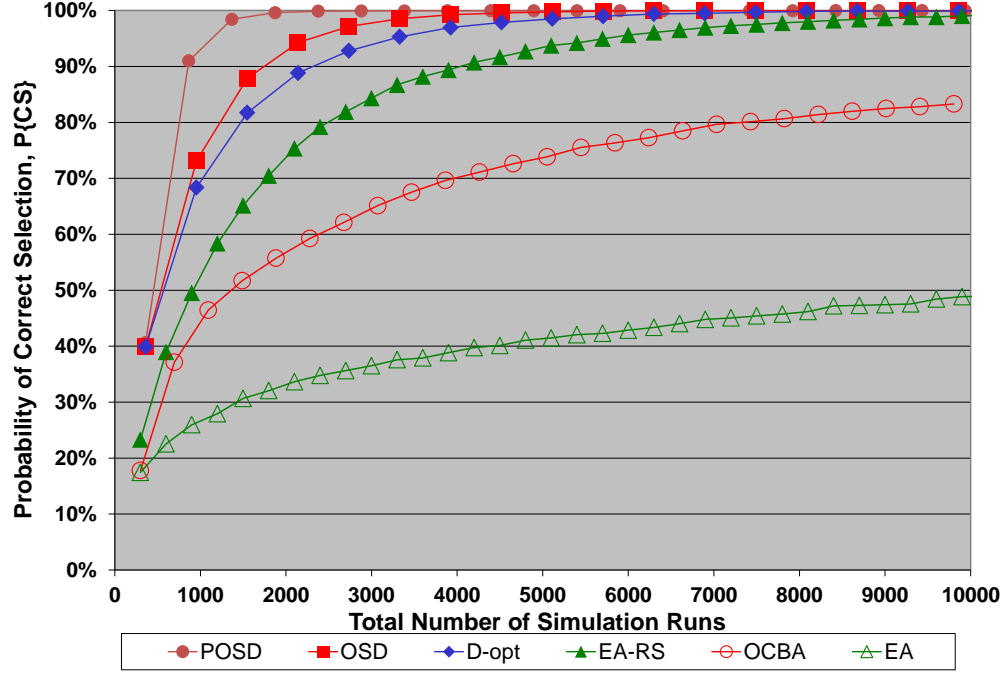
this function also has two local minima at  $x_6 \approx 3.42$  with  $y(x_6) \approx 0.16$  and  $x_{47} \approx 7.07$  with  $y(x_{47}) \approx -1.27$ .



**Figure 11:**  $y(x_i) = \sin(x_i) + \sin(10x_i/3) + \ln(x_i) - 0.84x_i + 3$

As mentioned above, we partitioned the domain for the regression based methods into six partitions and each of the local minimums are in a separate partition. Figure 12 shows the simulation results. POSD clearly performs the best since it uses a regression equation to capture the information and then efficiently allocates both between and within the partitions. The OSD and D-opt methods are the next best methods. They are regression based methods that at least allocate efficiently within the partitions. As a point of comparison, OSD achieves a 95% *PCS* after about 2,200 runs and D-optimal achieves the

same  $PCS$  after 3,300 runs. POSD achieves the same  $PCS$  after about 1,000 runs or about 45% of those required by OSD and 30% of those required by D-opt.



**Figure 12: Results of Experiment 7**

### Experiment 8 (Different noise distributions)

This experiment uses the same underlying function and domain used in Experiment 7 such that  $y(x_i) = \sin(x_i) + \sin(10x_i / 3) + \ln(x_i) - 0.84x_i + 3$ . We varied the type of the distribution for the noise terms of the simulation output while ensuring that each experiment used a distribution with a mean equal to zero and the variance equal to one. In addition to  $\varepsilon_h \sim N(0,1)$ , we also used:



- $\varepsilon_h \sim \text{Uniform}(\tau_1 = -\sqrt{3}, \tau_2 = \sqrt{3})$  where  $\tau_1$  and  $\tau_2$  are the lower and upper limits of the distribution,
- $\varepsilon_h \sim \text{Exponential}(\mu = 1) - 1$  where  $\mu$  is the mean of the distribution, and
- $\varepsilon_h \sim 2\text{Binomial}(\rho = 0.5) - 1$  where  $\rho$  is the probability of success of one trial.

The results of the experiment demonstrate that, for this problem, POSD is robust and performs relatively similar when assuming that the noise terms are normally distributed even if the noise terms are actually from one of the other three distributions. Table 4 below provides a sample of the results.

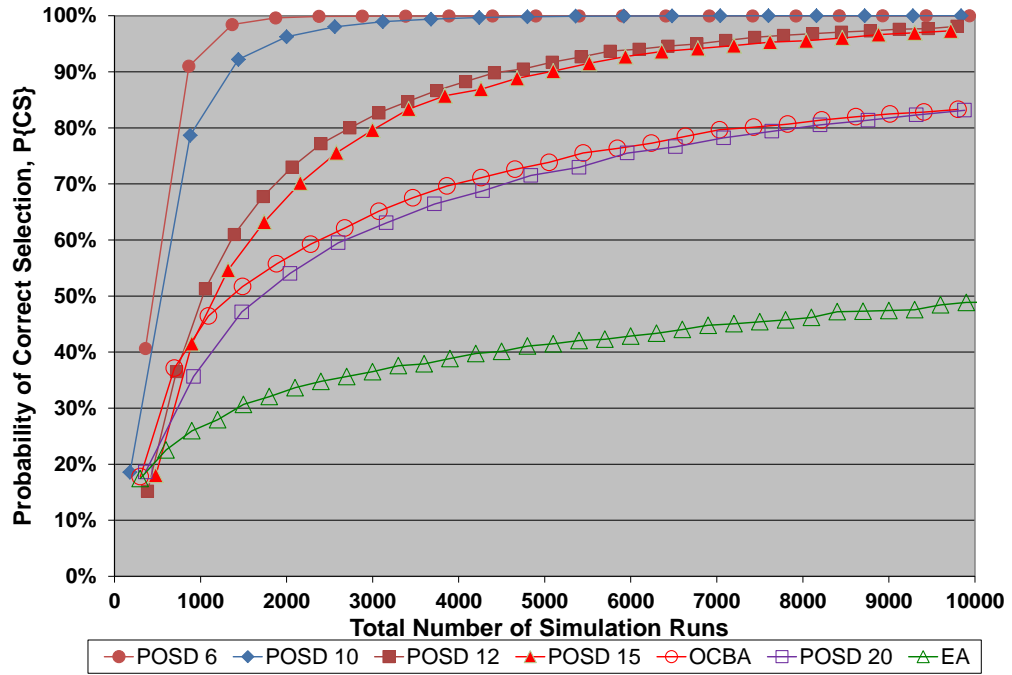
**Table 4: Results of Experiment 8**

Total Runs	PCS (Normal)	PCS (Uniform)	PCS (Exponential)	PCS (Binomial)
528	69.84%	70.06%	71.14%	70.70%
1032	95.34%	95.11%	95.06%	95.40%
1536	99.14%	99.09%	98.97%	99.19%
2040	99.76%	99.80%	99.51%	99.79%
2544	99.89%	99.93%	99.72%	99.88%

### Experiment 9 (Varied Number of Partitions)

This experiment also uses the same underlying function and domain used in Experiment 7 such that  $y(x_i) = \sin(x_i) + \sin(10x_i / 3) + \ln(x_i) - 0.84x_i + 3$ . We used POSD but varied the number of partitions used which also varied the location of the best global design within its partition. We experimented using six partitions, ten partitions, twelve partitions, fifteen partitions, and twenty partitions where the optimal design

location on the partitions are 7 out of 10, 3 out of 6, 2 out of 5, 3 out of 4, and 3 out of 3 respectively. The results of the experiment are in Figure 13 below and are similar to those from Experiment 6. POSD will do well if we can partition the domain so that the optimal design location is within the middle half of a partition and as close to the center of the partition as possible. Note the performance of POSD using ten partitions (POSD 10) and with the optimal design location in the center of the partition is almost as good as the performance of OSD using six partitions (POSD 6) where the design location is not near the middle of the partition. At the other extreme, in worst case scenarios when we have no good information to guide the partitioning or the underlying function is highly non-quadratic, the domain can be partitioned using sets of three design locations yielding an unbiased, piecewise quadratic estimate of the underlying function. Given partitions with only three design locations and a response surface that is fitted through the mean values at each design location across the entire domain, we expect that the performance will be similar to OCBA. Figure 13 shows that POSD with twenty partitions (POSD 20) and 3 design locations per partition performs slightly worse than OCBA after a small number of runs and gradually matches the performance of OCBA with a much larger computing budget.



**Figure 13: Results of Experiment 9**

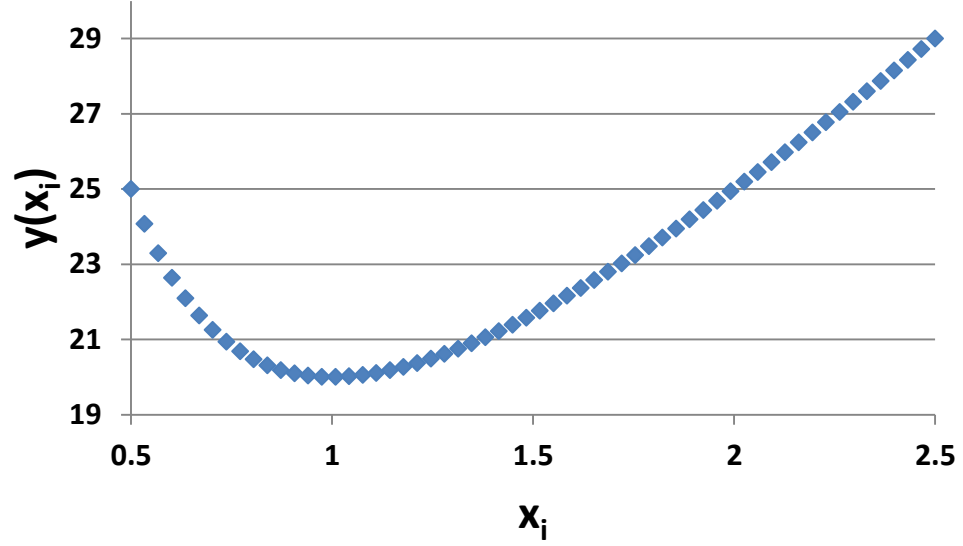
#### **Experiment 10 (one global minimum, asymmetric function, 60 design locations)**

We conducted this last experiment using the following function to represent the simulation output. Its skewed but convex nature is common among simple decision problems such as minimizing the cost of inventory problems or problems for the maximum concentration in the bloodstream for a single dose of a drug (Giordano and Weir, 1985).

$$f(x_i) = 10x_i + 10/x_i + N(0,1) .$$

We again used a domain consisting of 60 evenly spaced design locations where  $x \in [0.5, 2.5]$  such that, as shown in Figure 14 below, the global minimum is  $x_{16} \approx 1.01$

and  $y(x_{16}) \approx 20$ . As with the Experiment 7, we partitioned the domain for the regression based methods into six partitions.



**Figure 14:**  $y(x_i) = 10x_i + 10/x_i$

The results are consistent with Experiment 7 and are shown in Figure 15. D-optimal achieves an 89% *PCS* after about 10,000 and OSD achieves the same *PCS* after only 7,448 runs. POSD achieves an 89% *PCS* after about 2,600 runs or about 35% of those required by OSD and 26% of those required by D-opt.

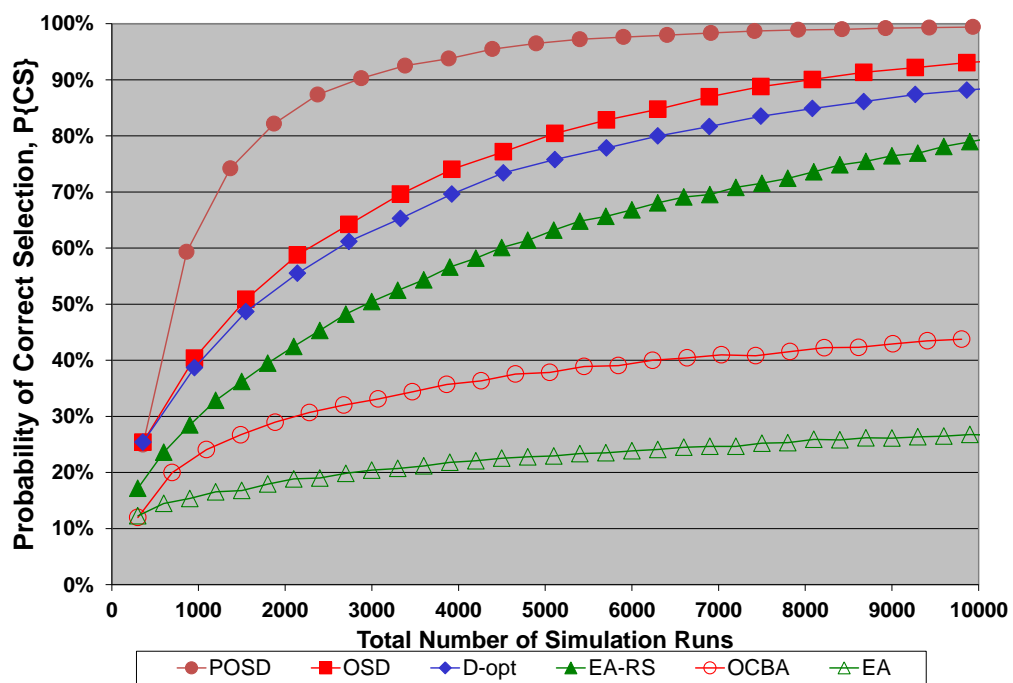


Figure 15: Results of Experiment 10

## CHAPTER 5 CONCLUSIONS AND FUTURE WORK

This dissertation explores the potential of enhancing R&S efficiency by incorporating simulation information from across the domain into a regression metamodel. We have developed an OSD method that can further enhance the efficiency of the simulation run allocation for selecting the best design. The OSD method offers approximately optimal rules that determine the design locations to conduct simulation runs and the number of samples allocated to each design location. Numerical experiments demonstrate that the use of a regression metamodel can indeed dramatically enhance simulation efficiency, even compared with some existing efficient R&S methods such as OCBA. As compared with methods using a regression metamodel, the OSD method offers a significant improvement over not only naïve response surface methods (by 50~70% reduction) but also the well known D-optimality criterion in DOE literature (by another 17%~27% reduction).

Though the use of a regression metamodel can dramatically enhance simulation efficiency, the regression-based methods, including our OSD, are constrained with some typical assumptions such as an underlining quadratic function for the means and homogeneous simulation noise. These assumptions can be alleviated if we can efficiently partition the domain so that we focus only on a small local area of the domain where the assumptions will hold. With this aim in mind, we have developed a POSD method for

selecting the best design on a partitioned domain. Our new method uses a heuristic based upon approximately optimal rules for between and within partitions that determine the number of samples allocated to each design location. Numerical experiments demonstrate that our new approach can dramatically enhance efficiency over existing efficient R&S methods.

There are certainly ways to expand this research and improve upon the OSD and POSD methods. Probably the area with the most potential is the integration of the POSD method with search or partitioning algorithms for general simulation optimization problems. As mentioned in Chapter 4, the partitions may be derived from knowledge of the domain, through iterative refinement such as a heuristic based upon the results of Experiment 6 in Chapter 3, or through an optimal selection procedure such as multivariate adaptive regression splines (MARS) (Friedman 1991). A heuristic based upon the latter two methods may prove promising. The MARS technique conducts a forward stepwise procedure to iteratively partition the domain into a piecewise polynomial spline. It conducts a forward stepwise procedure to pick the next partition boundary (knot location) that minimizes the error in fitting the model and then conducts a backwards stepwise procedure to remove knot locations to minimize the generalized cross validation (GCV). The forward stepwise procedure overfits the model and then the backwards step brings it back to a “reasonably” fit model. As opposed to the MARS technique, we are not necessarily interested in getting the best fit across the entire domain for our current simulation information. In the spirit of OCBA and OSD, intuitively we would like a good fit for portions of the domain that are critical in determining the best

design location and may be willing to accept a poor fit for the model in areas that are not critical to this decision.

Refining the allocation schemes or refining the implementation of the schemes of the OSD and POSD methods are other areas for future research. We used approximations and bounds to establish the allocation rules for both the OSD method and the POSD method. Each method may benefit by the use of different bounds or approximations. A way to improve the implementation of the methods would be the development of a heuristic to dynamically determine the number of runs to allocate during each iteration in order to mitigate the effects of the rounding rules. Brantley (2005, 2007) provides an improvement for the OCBA method that allocates one run at a time to reduce the numerical error in rounding the number of runs (Chen and Lee, 2010). While allocating only a single run during each iteration of POSD may be inefficient due to the symmetrical nature of the OSD within partition allocations, POSD may benefit from a similar dynamic rule that reduces the rounding when allocation between partitions.

This dissertation used very general assumptions that the underlying function was quadratic, that the noise terms are homoscedastic and normally distributed, and that the simulation run costs were equal across the entire decision domain. While partitioning of the domain may offset the impacts violating some of these assumptions, each of these assumptions also provides an area for further research. Possible extensions include incorporating the recent work of Yang (2010) that extends the de la Garza phenomenon to other nonlinear forms such as exponential and log-linear models. Yang's effort provides the minimum number of support points and the optimal locations for some of



the support points for these and other nonlinear forms. Dette and Melas (2010) extend the work of Yang to include a broader class of problems such as rational regression models (with polynomials for both the numerator and denominator).

Intuitively, since we are only sampling on three support points in each partition, we should be able to expand this method to a continuous domain by assuming that the number of design locations in each partition goes to infinity. However, we have to do some different treatments on the definition of *PCS* because the current one will go to zero as the number of design locations goes to infinity. In order to find the stationary point of the quadratic equation presented in (3) for a continuous domain, Melas et al. (2003) reformulate the linear (in the parameters) regression equation presented in (4) as a nonlinear regression equation and then solve for the extreme point of the new equation. Proposition 1 in Appendix F provides a connection between their work in the DOE community and the results for the interior design cases from (34) obtained from our *PCS* criterion. (In addition to the interior design cases, our method provides results for instances when the optimal solution is a boundary point and not necessarily a stationary point for the underlying equation.)

This dissertation focuses on one-dimensional problems and extending the method to higher dimensions is another area for further research. Morrice et al. (2008, 2009) extended the concepts from OSD to a method for selecting the best configuration (or design) based on a transient mean performance measure. The procedure extends the OCBA and OSD approaches to systems with means that are a function of some other variable such as time. Morrice et al. analyze the linear case and this prediction problem

can be viewed as a two dimensional POSD problem with each configuration representing a partition. For true multi-dimensional problems that are not just easily partitioned into one dimension segments, we propose that the OSD method can be combined with multi-dimensional search methods such as the Stochastic Trust Region Gradient-free Method (Chang, Hong, and Wan, 2007).

## APPENDICES

## APPENDIX A PROOF OF THEOREM 4

Proof: We can use the chain rule to establish that

$$\begin{aligned} \frac{\partial^2 Q}{\partial(\alpha_j)^2} &= \frac{-2}{\alpha_j^3} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] + \\ &\frac{1}{\alpha_j^2} \frac{\partial}{\partial \alpha_j} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] \end{aligned}$$

Using (28)

$$\begin{aligned} \frac{\partial^2 Q}{\partial(\alpha_j)^2} &= \frac{-2}{\alpha_j^3} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] + \\ &\frac{1}{\alpha_j^2} \left[ \left( \frac{-\sigma^2}{T} \frac{D_{A,j}^2}{\alpha_j^2} \right) \frac{\partial}{\partial \varsigma_A} \left( \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} \right) + \right. \\ &\left. \left( \frac{-\sigma^2}{T} \frac{D_{Z,j}^2}{\alpha_j^2} \right) \frac{\partial}{\partial \varsigma_Z} \left( \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right) \right] \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 Q}{\partial(\alpha_j)^2} &= \frac{-2}{\alpha_j^3} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] \\ &\frac{-\sigma^2}{T} \frac{1}{\alpha_j^4} \left[ \hat{d}(x_A) D_{A,j}^4 \frac{\partial}{\partial \varsigma_A} \left( \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{1}{(\varsigma_A)^{3/2}} \right) + \hat{d}(x_Z) D_{Z,j}^4 \frac{\partial}{\partial \varsigma_Z} \left( \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{1}{(\varsigma_Z)^{3/2}} \right) \right] \end{aligned}$$

$$\frac{\partial^2 \mathcal{Q}}{\partial (\alpha_j)^2} = \frac{-2}{\alpha_j^3} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right] \\ - \frac{\sigma^2}{T} \frac{1}{\alpha_j^4} \left[ \frac{\hat{d}(x_A) D_{A,j}^4}{2} \left( \frac{(\hat{d}(x_A))^2}{(\varsigma_A)^{7/2}} - \frac{3}{(\varsigma_A)^{5/2}} \right) \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \right. \\ \left. \frac{\hat{d}(x_Z) D_{Z,j}^4}{2} \left( \frac{(\hat{d}(x_Z))^2}{(\varsigma_Z)^{7/2}} - \frac{3}{(\varsigma_Z)^{5/2}} \right) \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \right] .$$

Combining terms

$$\frac{\partial^2 \mathcal{Q}}{\partial (\alpha_j)^2} = \left[ \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} - \frac{\sigma^2}{T} \frac{1}{\alpha_j^4} \frac{\hat{d}(x_A) D_{A,j}^4}{2} \left( \frac{(\hat{d}(x_A))^2}{(\varsigma_A)^{7/2}} - \frac{3}{(\varsigma_A)^{5/2}} \right) \right] \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \\ \left[ \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} - \frac{\sigma^2}{T} \frac{1}{\alpha_j^4} \frac{\hat{d}(x_Z) D_{Z,j}^4}{2} \left( \frac{(\hat{d}(x_Z))^2}{(\varsigma_Z)^{7/2}} - \frac{3}{(\varsigma_Z)^{5/2}} \right) \right] \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right)$$

$$\frac{\partial^2 \mathcal{Q}}{\partial (\alpha_j)^2} = \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} \left[ 1 + \frac{\sigma^2}{T} \frac{D_{A,j}^2}{4\alpha_j} \left( \frac{(\hat{d}(x_A))^2}{(\varsigma_A)^2} - \frac{3}{\varsigma_A} \right) \right] \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \\ \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \left[ 1 + \frac{\sigma^2}{T} \frac{D_{Z,j}^2}{4\alpha_j} \left( \frac{(\hat{d}(x_Z))^2}{(\varsigma_Z)^2} - \frac{3}{\varsigma_Z} \right) \right] \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right)$$

$$\frac{\partial^2 \mathcal{Q}}{\partial (\alpha_j)^2} = \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} \left[ 1 + \frac{\sigma^2}{T} \frac{3D_{A,j}^2}{4\alpha_j \varsigma_A} \left( \frac{(\hat{d}(x_A))^2}{3\varsigma_A} - 1 \right) \right] \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \\ \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \left[ 1 + \frac{\sigma^2}{T} \frac{3D_{Z,j}^2}{4\alpha_j \varsigma_Z} \left( \frac{(\hat{d}(x_Z))^2}{3\varsigma_Z} - 1 \right) \right] \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right)$$

Using (23)

$$\begin{aligned} \frac{\partial^2 Q}{\partial (\alpha_j)^2} &= \frac{-2}{\alpha_j^3} \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} \left[ 1 + \frac{3}{4} \frac{\frac{D_{A,j}^2}{\alpha_j}}{\left[ \frac{D_{A,l}^2}{\alpha_1} + \frac{D_{A,s}^2}{\alpha_s} + \frac{D_{A,k}^2}{\alpha_k} \right]} \left( \frac{(\hat{d}(x_A))^2}{3\varsigma_A} - 1 \right) \right] \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \\ &\frac{-2}{\alpha_j^3} \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \left[ 1 + \frac{3}{4} \frac{\frac{D_{Z,j}^2}{\alpha_j}}{\left[ \frac{D_{Z,l}^2}{\alpha_1} + \frac{D_{Z,s}^2}{\alpha_s} + \frac{D_{Z,k}^2}{\alpha_k} \right]} \left( \frac{(\hat{d}(x_Z))^2}{3\varsigma_Z} - 1 \right) \right] \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \end{aligned}$$

In the same manner

$$\frac{\partial^2 Q}{\partial \alpha_j \partial \alpha_i} = \frac{1}{\alpha_j^2} \frac{\partial}{\partial \alpha_i} \left[ \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{\hat{d}(x_A) D_{A,j}^2}{(\varsigma_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{\hat{d}(x_Z) D_{Z,j}^2}{(\varsigma_Z)^{3/2}} \right]$$

$$\begin{aligned} \frac{\partial^2 Q}{\partial \alpha_j \partial \alpha_i} &= \\ &\frac{-\sigma^2}{T} \frac{1}{\alpha_j^2 \alpha_i^2} \left[ \hat{d}(x_A) D_{A,j}^2 D_{A,i}^2 \frac{\partial}{\partial \varsigma_A} \left( \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) \frac{1}{(\varsigma_A)^{3/2}} \right) + \right. \\ &\left. \hat{d}(x_Z) D_{Z,j}^2 D_{Z,i}^2 \frac{\partial}{\partial \varsigma_Z} \left( \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \frac{1}{(\varsigma_Z)^{3/2}} \right) \right] \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 Q}{\partial \alpha_j \partial \alpha_i} &= \\ &\frac{-\sigma^2}{T} \frac{1}{\alpha_j^2 \alpha_i^2} \left[ \hat{d}(x_A) D_{A,j}^2 D_{A,i}^2 \left( \frac{(\hat{d}(x_A))^2}{(\varsigma_A)^{7/2}} - \frac{3}{(\varsigma_A)^{5/2}} \right) \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \right. \\ &\left. \hat{d}(x_Z) D_{Z,j}^2 D_{Z,i}^2 \left( \frac{(\hat{d}(x_Z))^2}{(\varsigma_Z)^{7/2}} - \frac{3}{(\varsigma_Z)^{5/2}} \right) \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \right] \end{aligned}$$

$$\frac{\partial^2 Q}{\partial \alpha_j \partial \alpha_i} =$$

$$-\frac{\sigma^2}{T} \frac{1}{\alpha_j^2 \alpha_i^2} \left[ \frac{3\hat{d}(x_A) D_{A,j}^2 D_{A,i}^2}{2(\varsigma_A)^{5/2}} \left( \frac{(\hat{d}(x_A))^2}{3\varsigma_A} - 1 \right) \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right) + \right.$$

$$\left. \frac{3\hat{d}(x_Z) D_{Z,j}^2 D_{Z,i}^2}{2(\varsigma_Z)^{5/2}} \left( \frac{(\hat{d}(x_Z))^2}{3\varsigma_Z} - 1 \right) \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right) \right]$$

$$\frac{\partial^2 Q}{\partial \alpha_j \partial \alpha_i} = -\frac{3\hat{d}(x_A)}{2(\varsigma_A)^{3/2}} \frac{\frac{D_{A,j}^2 D_{A,i}^2}{\alpha_j^2 \alpha_i^2}}{\left[ \frac{D_{A,l}^2}{\alpha_l} + \frac{D_{A,s}^2}{\alpha_s} + \frac{D_{A,k}^2}{\alpha_k} \right]} \left( \frac{(\hat{d}(x_A))^2}{3\varsigma_A} - 1 \right) \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\varsigma_A}}\right)$$

$$-\frac{3\hat{d}(x_Z)}{2(\varsigma_Z)^{3/2}} \frac{\frac{D_{Z,j}^2 D_{Z,i}^2}{\alpha_j^2 \alpha_i^2}}{\left[ \frac{D_{Z,l}^2}{\alpha_l} + \frac{D_{Z,s}^2}{\alpha_s} + \frac{D_{Z,k}^2}{\alpha_k} \right]} \left( \frac{(\hat{d}(x_Z))^2}{3\varsigma_Z} - 1 \right) \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\varsigma_Z}}\right)$$

Therefore, the Hessian matrix for the objective function can be written as:

$$\nabla^2 Q = \frac{-[\hat{d}(x_A)]^3}{2(\zeta_A)^{5/2}} \left[ \frac{D_{A,1}^2}{\alpha_1} + \frac{D_{A,s}^2}{\alpha_s} + \frac{D_{A,k}^2}{\alpha_k} \right] \begin{bmatrix} \frac{D_{A,1}^4}{\alpha_1^4} & \frac{D_{A,1}^2 D_{A,s}^2}{\alpha_1^2 \alpha_s^2} & \frac{D_{A,1}^2 D_{A,k}^2}{\alpha_1^2 \alpha_k^2} \\ \frac{D_{A,1}^2 D_{A,s}^2}{\alpha_1^2 \alpha_s^2} & \frac{D_{A,s}^4}{\alpha_s^4} & \frac{D_{A,s}^2 D_{A,k}^2}{\alpha_s^2 \alpha_k^2} \\ \frac{D_{A,1}^2 D_{A,k}^2}{\alpha_1^2 \alpha_k^2} & \frac{D_{A,s}^2 D_{A,k}^2}{\alpha_s^2 \alpha_k^2} & \frac{D_{A,k}^4}{\alpha_k^4} \end{bmatrix} \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\zeta_A}}\right) +$$

$$\frac{-[\hat{d}(x_Z)]^3}{2(\zeta_Z)^{5/2}} \left[ \frac{D_{Z,1}^2}{\alpha_1} + \frac{D_{Z,s}^2}{\alpha_s} + \frac{D_{Z,k}^2}{\alpha_k} \right] \begin{bmatrix} \frac{D_{Z,1}^4}{\alpha_1^4} & \frac{D_{Z,1}^2 D_{Z,s}^2}{\alpha_1^2 \alpha_s^2} & \frac{D_{Z,1}^2 D_{Z,k}^2}{\alpha_1^2 \alpha_k^2} \\ \frac{D_{Z,1}^2 D_{Z,s}^2}{\alpha_1^2 \alpha_s^2} & \frac{D_{Z,s}^4}{\alpha_s^4} & \frac{D_{Z,s}^2 D_{Z,k}^2}{\alpha_s^2 \alpha_k^2} \\ \frac{D_{Z,1}^2 D_{Z,k}^2}{\alpha_1^2 \alpha_k^2} & \frac{D_{Z,s}^2 D_{Z,k}^2}{\alpha_s^2 \alpha_k^2} & \frac{D_{Z,k}^4}{\alpha_k^4} \end{bmatrix} \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\zeta_Z}}\right)$$

We see that the Hessian matrix is negative semi-definite and conclude that the objective function (26) is concave. Therefore, the allocation rule given in (25) will yield a global (although not necessarily unique) optimal solution to (26) (Bazarra, et al., 1993 and Burden and Faires, 1993).



## APPENDIX B PROOF OF THEOREM 5

Proof: In order to simplify both the notation and derivation, we provide a derivation for a continuous choice of  $x_s$  and then apply the results to the discrete domain that is presented in Theorem 5.

We can use the chain rule to establish that

$$\frac{\partial Q}{\partial x_s} = - \frac{\partial Q}{\partial \varsigma_M} \frac{\partial \varsigma_M}{\partial x_s}.$$

Using the results from (27), we obtain

$$\frac{\partial Q}{\partial x_s} = - \phi\left(\frac{\hat{d}(x_M)}{\sqrt{\varsigma_M}}\right) \frac{\hat{d}(x_M)}{2(\varsigma_M)^{3/2}} \frac{\partial \varsigma_M}{\partial x_s}.$$

Substituting the results of (33) into (23), we obtain

$$\begin{aligned} \varsigma_M &= \frac{\sigma^2}{T} \left[ |D_{M,l}| + |D_{M,s}| + |D_{M,k}| \right]^2 \\ \frac{\partial \varsigma_M}{\partial x_s} &= \frac{2\sigma^2}{T} \left[ |D_{M,l}| + |D_{M,s}| + |D_{M,k}| \right] \left[ \frac{\partial |D_{M,l}|}{\partial x_s} + \frac{\partial |D_{M,s}|}{\partial x_s} + \frac{\partial |D_{M,k}|}{\partial x_s} \right]. \end{aligned} \tag{B1}$$

Define the sign function as  $\mathbf{sgn}(w) = \frac{w}{\text{abs}(w)}$  when  $w \neq 0$ . This function is undefined

when  $w = 0$ . Using this notation, we can rewrite (B1) as

$$\begin{aligned} \frac{\partial \zeta_M}{\partial x_s} &= \frac{2\sigma^2}{T} \left[ |D_{M,1}| + |D_{M,s}| + |D_{M,k}| \right] \\ &\left[ \mathbf{sgn}(D_{M,1}) \frac{\partial D_{M,1}}{\partial x_s} + \mathbf{sgn}(D_{M,2}) \frac{\partial D_{M,2}}{\partial x_s} + \mathbf{sgn}(D_{M,3}) \frac{\partial D_{M,3}}{\partial x_s} \right] \end{aligned} \quad (\text{B2})$$

where

$$\begin{aligned} \frac{\partial D_{M,1}}{\partial x_s} &= \frac{(x_M - x_b)(x_M + x_b - x_1 - x_k)}{(x_1 - x_s)^2 (x_1 - x_k)} \\ \frac{\partial D_{M,2}}{\partial x_s} &= \frac{-(x_M - x_b)(x_M + x_b - x_1 - x_k)(2x_s - x_1 - x_k)}{(x_s - x_1)^2 (x_s - x_k)^2} \\ \frac{\partial D_{M,3}}{\partial x_s} &= \frac{(x_M - x_b)(x_M + x_b - x_1 - x_k)}{(x_k - x_1)(x_k - x_s)^2} . \end{aligned}$$

To determine the optimal location for  $x_s$ , we must consider five cases which, as will be

shown, result from the combinations of  $\mathbf{sgn}(x_M + x_b - x_1 - x_k) = \{-1, 0, 1\}$  and

$\mathbf{sgn}(2x_s - x_1 - x_k) = \{-1, 1\}$ .

Case I:  $\frac{x_M + x_b}{2} < \frac{3x_1 + x_k}{4}$ . For this case,

$$\mathbf{sgn}(D_{M,1}) = -\mathbf{sgn}(x_M - x_b)$$

$$\mathbf{sgn}(D_{M,2}) = \mathbf{sgn}(x_M - x_b)$$

$$\mathbf{sgn}(D_{M,3}) = \mathbf{sgn}(x_M - x_b) \cdot \mathbf{sgn}(x_M + x_b - x_1 - x_s).$$

Substituting these results and the fact that  $\mathbf{sgn}(x_M + x_b - x_1 - x_s) = 1$  when

$x_s < x_M + x_b - x_1$ , we obtain

$$\frac{\partial \zeta_M}{\partial x_s} = \frac{4\sigma^2}{T} \left[ |D_{M,l}| + |D_{M,s}| + |D_{M,k}| \right] \cdot |x_M - x_b| \frac{(x_M + x_b - x_1 - x_k)}{(x_1 - x_s)^2 (x_k - x_1)} < 0$$

or

$$\frac{\partial Q}{\partial x_s} > 0 \text{ when } x_s < x_M + x_b - x_1.$$

In the same manner,  $\mathbf{sgn}(x_M + x_b - x_1 - x_s) = -1$  when  $x_s > x_M + x_b - x_1$ . Substituting into (B2)

$$\frac{\partial \zeta_M}{\partial x_s} = \frac{4\sigma^2}{T} \left[ |D_{M,l}| + |D_{M,s}| + |D_{M,k}| \right] \cdot |x_M - x_b| \frac{(x_M + x_b - x_1 - x_k)(2x_s - x_1 - x_k)}{(x_1 - x_s)^2 (x_k - x_s)^2}.$$

Analyzing the above result, we see that  $\frac{\partial Q}{\partial x_s} > 0$  when  $x_s < (x_1 + x_k)/2$ ;  $\frac{\partial Q}{\partial x_s} = 0$  when

$x_s = (x_1 + x_k)/2$ ; and  $\frac{\partial Q}{\partial x_s} < 0$  when  $x_s > (x_1 + x_k)/2$ . Therefore, when

$\frac{x_M + x_b}{2} < \frac{3x_1 + x_k}{4}$ , we choose  $x_s = (x_1 + x_k)/2$ . Note that  $\frac{\partial \zeta_M}{\partial x_s}$  is not defined when

$x_s = x_M + x_b - x_1$ . However,  $\zeta_M$  is continuous at this location and the results above

demonstrate that  $\frac{\partial Q}{\partial x_s}$  is positive as we approach the location from the positive and

negative directions.

Case II:  $\frac{3x_1 + x_k}{4} \leq \frac{x_M + x_b}{2} < \frac{x_1 + x_k}{2}$ .

We can use the results for Case I to show that  $\frac{\partial Q}{\partial x_s} > 0$  when  $x_s < x_M + x_b - x_1$ .

When  $[(3x_1 + x_k)/2 \leq x_M + x_b < x_1 + x_k] \cap [x_s > x_M + x_b - x_1]$ , we can establish that

$(2x_s - x_1 - x_k) > 0$  and  $\frac{\partial Q}{\partial x_s} < 0$ . Since  $\frac{\partial Q}{\partial x_s} > 0$  when  $x_s < x_M + x_b - x_1$  and  $\frac{\partial Q}{\partial x_s} < 0$  when

$x_s > x_M + x_b - x_1$ , we choose  $x_s = x_M + x_b - x_1$ . As in Case I,  $\frac{\partial \varsigma_M}{\partial x_s}$  is not defined when

$x_s = x_M + x_b - x_1$ . However,  $\varsigma_M$  is continuous at this location and the results above

demonstrate that  $\frac{\partial Q}{\partial x_s}$  is positive as we approach the location from the negative direction

and negative as we approach the location from the positive direction. Also, note that as

$$\frac{x_M + x_b}{2} \rightarrow \frac{x_1 + x_k}{2}, x_s \rightarrow x_k.$$

Case III:  $\frac{x_M + x_b}{2} > \frac{x_1 + 3x_k}{4}$ . For this case, we know that

$$\mathbf{sgn}(D_{M,1}) = \mathbf{sgn}(x_M - x_b) \cdot \mathbf{sgn}(x_M + x_b - x_s - x_k)$$

$$\mathbf{sgn}(D_{M,2}) = -\mathbf{sgn}(x_M - x_b)$$

$$\mathbf{sgn}(D_{M,3}) = \mathbf{sgn}(x_M - x_b)$$

The remainder of the proof for this case is similar to the proof for Case I and we omit the details for brevity.

Case IV:  $\frac{x_1 + x_k}{2} < \frac{x_M + x_b}{2} \leq \frac{x_1 + 3x_k}{4}$ . The proof for this case is similar to the proof for

Case II. We again omit the details for brevity.

Case V:  $\frac{x_1 + x_k}{2} = \frac{x_M + x_b}{2}$ . The derivative is not defined for this case. However, using

the results from Case II and Case IV, we know that  $x_s \rightarrow x_1$  as  $\frac{x_M + x_b}{2} \rightarrow \frac{x_1 + x_k}{2}$  from

the negative side and  $x_s \rightarrow x_1$  as  $\frac{x_M + x_b}{2} \rightarrow \frac{x_1 + x_k}{2}$  from the positive side. The extreme

points cover these two cases. Therefore, we can choose  $x_s$  at any location on the domain

and will choose  $x_s = (x_1 + x_k)/2$  for consistency with Case I and Case III.

The results from Cases I – V demonstrate that:

$$x_s = \begin{cases} x_M + x_b - x_1, & \frac{3x_1 + x_k}{4} \leq \frac{x_M + x_b}{2} < \frac{x_1 + x_k}{2} \\ x_M + x_b - x_k, & \frac{x_1 + x_k}{2} < \frac{x_M + x_b}{2} \leq \frac{x_1 + 3x_k}{4} \\ (x_1 + x_k)/2, & otherwise \end{cases} \quad (B3)$$

Note that the solutions in (B3) are derived from an examination on a continuous domain

and the approximate PCS expressed in (18) with allocations that satisfy (33) is

maximized by selecting the interior design location closest to  $x_s$  selected from (B3).

When presented with design locations evenly spaced across the domain such that

$$\Delta = x_{i+1} - x_i = \frac{x_k - x_1}{k-1} \quad \forall i = 1, \dots, k-1, \text{ we can show for the first case in (B3) that}$$

$$x_M + x_b - x_1 = x_1 + (M-1)\Delta + x_1 + (b-1)\Delta - x_1 = x_1 + [(M+b-1)-1]\Delta = x_{M+b-1}.$$

The second cases in (B3) can be proven in a similar manner. When presented with an odd number of design locations evenly spaced across the domain, the third case in (B3) can also be proven such that

$$x_s = \begin{cases} x_{M+b-1}, & \frac{3x_1 + x_k}{4} \leq \frac{x_M + x_b}{2} < \frac{x_1 + x_k}{2} \\ x_{M+b-k}, & \frac{x_1 + x_k}{2} < \frac{x_M + x_b}{2} \leq \frac{x_1 + 3x_k}{4} \\ x_{(k-1)/2}, & \text{otherwise} \end{cases}$$

## APPENDIX C PROMODEL CODE FOR OSD (ONE PARTITION CASE)

See [www.promodel.com](http://www.promodel.com)

```

*****
*
*
*           Formatted Listing of Model:           *
* C:\Users\Admin\Documents\GMU\Research\Dissertation prep info\Promodel
code\Discrete with Response Surface v10.1 contour map.MOD *
*
*****

Time Units:           Minutes
Distance Units:       Feet
*****
*           Locations           *
*****
Name      Cap   Units Stats   Rules   Cost
-----
Experiment_loc 1    1   None   Oldest, ,
Run_Queue   Infinite 1   None   Oldest, ,
Run_Allocation 1    1   None   Oldest, ,
*****
*           Entities           *
*****
Name      Speed (fpm) Stats   Cost
-----
Experiment 150      None
Run       150      None
*****
*           Processing         *
*****
Entity   Location   Process           Routing
          Operation   Blk Output   Destination   Rule   Move Logic
-----
Experiment Experiment_loc INC v_iteration

a_runtotal = 0
v_designs = m_designs

```

```

A_RESSTATS[1] = 0
A_RESSTATS[2] = 0
A_RESSTATS[3] = 0
A_RESSTATS[4] = 0
A_RESSTATS[5] = 0
A_RESSTATS[6] = 0
A_RESSTATS[7] = 0
A_RESSTATS[8] = 0
A_RESSTATS[9] = 0
A_RESSTATS[10] = 0
A_RESSTATS[11] = 0
A_RESSTATS[12] = 0
A_RESSTATS[13] = 0
A_RESSTATS[14] = 0
A_RESSTATS[15] = 0
A_RESSTATS[16] = 0
A_RESSTATS[17] = 0
A_RESSTATS[18] = 0
A_RESSTATS[19] = 0
A_RESSTATS[20] = m_designs
A_RESSTATS[21] = 0
A_RESSTATS[22] = 0

```

```

INT TEMPCOUNT
TEMPCOUNT = 1
WHILE TEMPCOUNT < (m_designs+1) DO
BEGIN
  a_RunCount[TEMPCOUNT,1] = 0
  a_RunCount[TEMPCOUNT,2] = 0

  A_Design_STATS[TEMPCOUNT,1] = 0
  A_Design_STATS[TEMPCOUNT,2] = 0
  A_Design_STATS[TEMPCOUNT,3] = 0
  A_Design_STATS[TEMPCOUNT,4] = 0
  A_Design_STATS[TEMPCOUNT,5] = 0
  A_Design_STATS[TEMPCOUNT,6] = 0
  A_Design_STATS[TEMPCOUNT,7] = 0
  A_Design_STATS[TEMPCOUNT,8] = 0
  A_Design_STATS[TEMPCOUNT,9] = 0
  A_Design_STATS[TEMPCOUNT,10] = 0

  a_error_factor[tempcount] = U(1,1)

```



```

        INC TEMPCOUNT
    END

    Interval = Real(m_Maxx-m_Minx)/Real(m_designs-1)

# This randomly generates optimal solution

# ***** for x^2 and binomial experiment
    # V_SOLUTION = U( (m_Maxx+m_Minx)/2 , (m_Maxx-m_Minx)/2 )
# This fixes the optimal solution
    V_SOLUTION = 3.0/3.0

# This block is for the Experiment with c0, c1, c2 ~ U(-10,10)
    #v_c0 = U(0,10)
    #v_c1 = U(0,10)
    #v_c2 = U(0,10)
    #V_SOLUTION = -0.5*v_c1/v_c2

# ***** for x^4
    # V_SOLUTION = U( (m_Maxx+m_Minx)/2 , (m_Maxx-m_Minx)/4 )

# *****For fixed solution
    # V_SOLUTION = (m_Maxx+m_Minx)/2

# This block finds the design with the lowest mean.
    INT bestloop
    REAL distbest, tempint, disttemp

    v_best = 1
    tempint = 0

# ***** for x^2 and binomial experiment
    distbest = (m_minx+tempint-v_solution)*(m_minx+tempint-v_solution)

    bestloop = 2
    WHILE bestloop < (v_designs+1) DO
        BEGIN
            tempint = tempint + interval

# ***** for x^2 and binomial experiment
            disttemp = (m_minx+tempint-v_solution)*(m_minx+tempint-v_solution)

            IF disttemp < distbest Then
                Begin

```

```

        distbest = disttemp
        v_best = bestloop
    End
    INC bestloop
END

```

```

V_EXPERIMENTRUN = 0
v_RUNPER = 0

```

```

TEMPCOUNT = 1
WHILE TEMPCOUNT < (m_increments+1) DO
BEGIN
    INC a_runtotal
    ORDER 1 RUN TO Run_Queue
    Wait 0.4 sec
    INC TEMPCOUNT
END

```

```

                                1  Experiment EXIT      FIRST 1
Run    Run_Queue    Wait 0.1 sec    1  Run    Run_Allocation FIRST 1
Run    Run_Allocation INT TEMPQUEUE, I, J, I2, J2,IOUT

```

```

m_MAIN

```

```

        Wait 1 sec        1  Run    EXIT      FIRST 1

```

```

*****
*                               *
*               Arrivals        *
*****

```

Entity	Location	Qty Each	First Time	Occurrences	Frequency	Logic
Experiment	Experiment_loc	1	0 sec	10000	60 min	

```

*****
*                               *
*               Attributes        *
*****

```

ID	Type	Classification
a_runtotal	Integer	Entity

```

*****
*                               *
*               Variables (global)        *

```

\*\*\*\*\*

ID	Type	Initial value	Stats
ylowest	Real	0	None
xlowest	Real	0	None
v_designs	Integer	m_designs	None
v_SOLUTION	Real	0	None
v_local1	Real	0	None
v_local2	Real	0	None
v_LOWMEAN	Real	0	None
v_LowX	Real	0	None
v_LOWEST	Integer	0	None
v_RunPer	Integer	0	None
v_ExperimentRun	Integer	0	None
v_TotalRun	Integer	0	None
V_STAND_VAR	Real	0	None
TempLoop	Integer	0	None
TempRun	Real	0	None
v_Next_C_Run	Integer	0	None
interval	Real	0	None
v_best	Integer	0	None
v_iteration	Integer	0	None
v_varbeta	Real	0	None
v_xc	Real	0	None
v_rho1	Real	0	None
v_rho2	Real	0	None
v_z1	Real	0	None
v_z2	Real	0	None
v_z3	Real	0	None
v_del_bminus	Real	0	None
v_del_bplus	Real	0	None
v_b	Real	0	None
v_B2	Real	0	None
v_B2_actual	Integer	0	None
v_c0	Real	0	None
v_c1	Real	0	None
v_c2	Real	0	None
v_support	Integer	0	None

\*\*\*\*\*

### Arrays \*

\*\*\*\*\*

ID	Dimensions	Type	Import File	Export File	Disable	Persist
----	------------	------	-------------	-------------	---------	---------

#COL 1: Location of X  
 #COL 2: Sum of X  
 #COL 3: Sum of X^2  
 #COL 4: Sum of X^3  
 #COL 5: Sum of X^4  
 #COL 6: Sum of y  
 #COL 7: Sum of yx  
 #COL 8: Sum of yx^2  
 #COL 9: Sum of y^2  
 #COL 10: Variance of runs for this design  
 #COL 11: Predicted response at each location  
 #COL 12: Delta b,i  
 #COL 13: Li,1  
 #COL 14: Li,2  
 #COL 15: Li,3  
 #COL 16: Sigma^2\_b,i  
 #COL 17: phi\_i  
 #COL 18: denominator for secant line  
 a\_Design\_stats 110,18      Real                      None              No  
 a\_RunCount    110,2      Integer                      None              No  
 #  
 #COL 1: Mean of Left Design  
 #COL 2: Mean of Middle Design  
 #COL 3: Mean of Right Design  
 #COL 4: beta2 term  
 #COL 5: beta1 term  
 #COL 6: beta0 term  
 #COL 7: x\*  
 #COL 8: y\*  
 #COL 9: lowest y  
 #COL 10: lowest x  
 #COL 11:  
 #COL 12: Sum of Runs  
 #COL 13: Variance of optimal response  
 #COL 14: OCBA Coefficient  
 #COL 15: OCBA Run Allocation  
 #COL 16: Delta term for OCBA  
 #COL 17: If x\* is optimal, then this is set to 1 (used for v&v only)  
 #COL 18:  
 #COL 19:  
 #COL 20: Width of Partition (number of designs)  
 #COL 21: Xleft  
 #COL 22: Xright  
 #

```

#COL 24: A1 (for c-opt)
#COL 25: A2
#COL 26: A3
#COL 27: alpha 1
#COL 28: alpha 2
#COL 29: alpha 3
#
#COL 31: Inverse 1,1
#COL 32: Inverse 1,2
#COL 33: Inverse 1,3
#COL 34: Inverse 2,1
#COL 35: Inverse 2,2
#COL 36: Inverse 2,3
#COL 37: Inverse 3,1
#COL 38: Inverse 3,2
#COL 39: Inverse 3,3
#
a_ResStats  39      Real                None      No
#
#COL 1: P{CS}
#COL 2: Mean Distance from local
#COL 3: Squared Distance from local
#COL 4: Mean y_hat(x*)
#COL 5: Squared y_hat(x*)
#COL 6: Mean f(x*)
#COL 7: Squared f(x*)
#COL 8: rho1
#COL 9: rho2
#COL 10: Allocations to Design 1
#COL 20: Allocations to Design 11
a_Results  1501,110  Real                a_Results.xls None      No
#
#COL 1: Coefficients
#COL 2: Temp Column
a_B        3,3      Real                None      No
a_RHS      3        Real                None      No
a_INV      3,3      Real                None      No
FTF_MATRIX 3,3      Real                None      No
a_TEMP     6,6      Real                None      No
a_error_factor 110  Real                None      No
a_F        3        Real                None      No
a_E        3        Real                None      No
a_alpha    3        Real                None      No

```

```

*****
*                                     *
*                               Macros                               *
*                                     *
*****

```

ID	Text
m_nzero	2
m_initial_loops	2
m_delta	14
m_restricted	1
m_MAIN	

```

# This block sets the number of runs for each partition to the value of nzero (initial
iterations)

        IF A_RUNTOTAL < m_initial_loops THEN
        BEGIN
        #   M_INITIAL_RUNS
# Conducts 2 iterations of D-opt instead of runs to entire domain
        m_D_opt
        m_D_opt

        m_MESH_CONSTRUCTION
        END

# This block does the subsequent iterations.

        ELSE
        BEGIN
        m_optimality_criteria
        END

# This checks to see how many total runs a design has been allocated
# and how many it currently has. If it needs more, it calls the macro for
# allocating a new run based upon the simulated underlying distribution.

        m_runs_discrete

#This block allows us to generate performance statistics for different computing budgets.

        m_response_stats

        m_pcs_calculation

m_optimality_criteria

```

```

# Choose the optimality criteria
# Use for D-opt
      # m_D_opt

# Use for Equal Allocation - Response Surface
      # M_INITIAL_RUNS

# Use for OSD
      transient_PCS_opt

      m_response_stats
# Generates a response surface across the entire partition

      REAL AMAX, SWITCH, TEMPCOE, TEMPDIF
      INT NMAX, II, JJ, III, IOUTLOOP
      INT IJKL,IMN
      REAL TOL, TEMPTOL

      m_Info_matrix

# Determines the total runs allocated for the entire partition and the response at each
discrete point
      INT IIJJ
      IIJJ = 1
      ylowest =
a_Resstats[6]+a_Resstats[5]*a_design_stats[IIJJ,1]+a_Resstats[4]*a_design_stats[IIJJ,1]
**2
      v_lowest = 1
      WHILE IIJJ < (m_designs+1) DO
      BEGIN
        a_Resstats[12] = a_Resstats[12] + Real(a_RunCount[IIJJ,2])
        a_design_stats[IIJJ,11]=
a_Resstats[6]+a_Resstats[5]*a_design_stats[IIJJ,1]+a_Resstats[4]*a_design_stats[IIJJ,1]
**2
        IF a_design_stats[IIJJ,11] < ylowest then
        BEGIN
          ylowest = a_design_stats[IIJJ,11]
          v_lowest = IIJJ
        END
      INC IIJJ
      END

#####
# Use this for OSD

```

```

                                m_Diff_opt_response
#####

m_d_opt_runs          1

Experiment_Parameters *****
m_minx                -1
m_maxx                1
m_designs             11
m_increments          1+(1100 - m_designs * m_nzero)/m_delta
NDIM                  3
One_Partition_Code    *****
m_initial_runs
# (Generic) This block sets the number of runs for each partition to the value of nzero
(initial iterations)
    TEMPQUEUE = 1
    WHILE TEMPQUEUE < (v_designs+1) DO
    BEGIN
        INC a_RunCount[TEMPQUEUE,1], m_nzero
        INC TEMPQUEUE
    END

m_MESH_CONSTRUCTION
# This block checks to see how many total runs a design has been allocated
# and how many it currently has. If it needs more, it calls the macro for
# allocating a new run based upon the simulated underlying distribution.

    INT Tempmesh
    Tempmesh = 1
    While Tempmesh < (v_designs+1) DO
    Begin
        a_Design_stats[Tempmesh,1]= m_Minx + Real(Tempmesh-
1)*Real(m_Maxx-m_Minx)/Real(m_designs-1)
        INC Tempmesh
    End

m_runs_discrete
# This block checks to see how many total runs a design has been allocated
# and how many it currently has. If it needs more, it calls the macro for
# allocating a new run based upon the simulated underlying distribution.

    Temploop = 1
    While Temploop < (v_designs+1) DO
    Begin
        While a_RunCount[Temploop,2] < a_RunCount[Temploop,1] DO

```



```

Begin
  m_run_generation
  A_Design_STATS[Temploop,2] =
A_Design_STATS[Temploop,2] + a_design_stats[Temploop,1]
  A_Design_STATS[Temploop,3] =
A_Design_STATS[Temploop,3] + a_design_stats[Temploop,1]**2
  A_Design_STATS[Temploop,4] =
A_Design_STATS[Temploop,4] + a_design_stats[Temploop,1]**3
  A_Design_STATS[Temploop,5] =
A_Design_STATS[Temploop,5] + a_design_stats[Temploop,1]**4
  A_Design_STATS[Temploop,6] =
A_Design_STATS[Temploop,6] + RunValue
  A_Design_STATS[Temploop,7] =
A_Design_STATS[Temploop,7] + RunValue*a_design_stats[Temploop,1]
  A_Design_STATS[Temploop,8] =
A_Design_STATS[Temploop,8] + RunValue*a_design_stats[Temploop,1]**2
  A_Design_STATS[Temploop,9] =
A_Design_STATS[Temploop,9] + RunValue**2
  INC a_RunCount[Temploop,2]
End

  INC Temploop
End

m_run_generation
# This macro contains the specific formulae for each experiment
  INC V_totalrun
  INC v_ExperimentRun
  INC a_results[a_runtotal,Temploop+9]

# This block allocates the run

  REAL RunValue

#####Experiment 1

# used for homoscedastic
  RunValue = (a_design_stats[Temploop,1]-v_SOLUTION)**2 + N(0,1)

  ***** for -x^2
  #RunValue = -((a_design_stats[Temploop,1]-v_SOLUTION)**2) + N(0,1)

# normal but heteroscedastic

```

```

# RunValue = (a_design_stats[Temploop,1]-v_SOLUTION)**2 +
a_error_factor[temploop]*N(0,4)
#***** for new experiment 1
# RunValue = v_c2*(a_design_stats[Temploop,1])**2 +
v_c1*(a_design_stats[Temploop,1]) + v_c0 + N(0,1)

# for binomial experiment

#RunValue = bi( 1 , ( 0.25*(a_design_stats[Temploop,1]-
v_SOLUTION)**2) )

# used for x^4
# RunValue = (a_design_stats[Temploop,1]-v_SOLUTION)**4 +
N(0,1)

m_Info_matrix
# This block is used to build the XTX Info matrix
II = 1
While II < (NDIM+1) DO
Begin
JJ = 1
a_RHS[II] = 0
While JJ < (NDIM+1) DO
Begin
FTF_MATRIX[JJ,II] = 0
INC JJ
END
INC II
END

II = 1
WHILE II < (m_designs + 1) DO
BEGIN
FTF_MATRIX[1,1] = FTF_MATRIX[1,1] + a_RunCount[II,2]
FTF_MATRIX[1,2] = FTF_MATRIX[1,2] +
A_Design_STATS[II,2]
FTF_MATRIX[1,3] = FTF_MATRIX[1,3] +
A_Design_STATS[II,3]
FTF_MATRIX[2,1] = FTF_MATRIX[2,1] +
A_Design_STATS[II,2]
FTF_MATRIX[2,2] = FTF_MATRIX[2,2] +
A_Design_STATS[II,3]
FTF_MATRIX[2,3] = FTF_MATRIX[2,3] +
A_Design_STATS[II,4]

```

```

      FTF_MATRIX[3,1] = FTF_MATRIX[3,1] +
A_Design_STATS[II,3]
      FTF_MATRIX[3,2] = FTF_MATRIX[3,2] +
A_Design_STATS[II,4]
      FTF_MATRIX[3,3] = FTF_MATRIX[3,3] +
A_Design_STATS[II,5]
      a_RHS[1] = a_RHS[1] + A_Design_STATS[II,6]
      a_RHS[2] = a_RHS[2] + A_Design_STATS[II,7]
      a_RHS[3] = a_RHS[3] + A_Design_STATS[II,8]

```

```

      INC II
    END

```

# This block is used to invert the XTX info matrix

```

    II = 1
    While II < (NDIM+1) DO
      Begin
        JJ = 1
        While JJ < (NDIM+1) DO
          Begin
            A_TEMP[JJ,II] = FTF_MATRIX[JJ,II]
            INC JJ
          END
        END
        INC II
      END
    END

```

```

    II = 1
    While II < (NDIM+1) DO
      Begin
        JJ = (NDIM+1)
        While JJ < (2*NDIM+1) DO
          Begin
            A_TEMP[II,JJ] = 0
            INC JJ
          END
        END
        A_TEMP[II,NDIM+II] = 1
        INC II
      END
    END

```

# This loop row reduces the first N columns of the matrix

```

    II = 1
    While II < (NDIM+1) DO

```

Begin

# This loop does maximum element pivoting for each column and then  
# divides the row by the first element in the row

```
      AMAX = A_TEMP[II,II]
      NMAX = II

      JJ = II+1
      WHILE JJ < (NDIM+1) DO
      BEGIN
        IF SQRT(A_TEMP[JJ,II]**2) > SQRT(AMAX**2) THEN
        BEGIN
          AMAX = A_TEMP[JJ,II]
          NMAX = JJ
        END
        INC JJ
      END

      IF II <> NMAX THEN
      BEGIN
        JJ = II
        WHILE JJ < (2*NDIM+1) DO
        BEGIN
          SWITCH = A_TEMP[II,JJ]
          A_TEMP[II,JJ] = A_TEMP[NMAX,JJ]/AMAX
          A_TEMP[NMAX,JJ] = SWITCH
        END
        INC JJ
      END
      ELSE
      BEGIN
        JJ = 1
        WHILE JJ < (2*NDIM+1) DO
        BEGIN
          A_TEMP[II,JJ] = A_TEMP[II,JJ]/AMAX
        END
        INC JJ
      END
    END
```

# This loop reduces all other elements in the column to zero

```
      JJ = 1
      WHILE JJ < (NDIM+1) DO
```

```

      BEGIN
      IF JJ <> II THEN
      BEGIN
      TEMPCOEF = -A_TEMP[JJ,II]
      III = II
      WHILE III < (2*NDIM+1) DO
      BEGIN
      A_TEMP[JJ,III] = A_TEMP[JJ,III] +
TEMPCOEF*A_TEMP[II,III]
      INC III
      END
      END
      INC JJ
      END

```

```

      INC II
      END

```

# This loop establishes the inverse matrix using the non-reduced rows

```

      II = 1
      WHILE II < (NDIM+1) DO
      BEGIN
      JJ = (NDIM+1)
      WHILE JJ < (2*NDIM+1) DO
      BEGIN
      A_INV[II,JJ-NDIM] = A_TEMP[II,JJ]
      INC JJ
      END
      INC II
      END

```

# This loop establishes the inverse matrix using the non-reduced rows

```

      a_RESstats[6] = a_INV[1,1]*a_RHS[1] + a_INV[1,2]*a_RHS[2] +
a_INV[1,3]*a_RHS[3]
      a_RESstats[5] = a_INV[2,1]*a_RHS[1] + a_INV[2,2]*a_RHS[2] +
a_INV[2,3]*a_RHS[3]
      a_RESstats[4] = a_INV[3,1]*a_RHS[1] + a_INV[3,2]*a_RHS[2] +
a_INV[3,3]*a_RHS[3]

      a_RESstats[31] = a_INV[1,1]
      a_RESstats[32] = a_INV[1,2]

```

```

a_RESstats[33] = a_INV[1,3]
a_RESstats[34] = a_INV[2,1]
a_RESstats[35] = a_INV[2,2]
a_RESstats[36] = a_INV[2,3]
a_RESstats[37] = a_INV[3,1]
a_RESstats[38] = a_INV[3,2]
a_RESstats[39] = a_INV[3,3]

```

```

Allocation_Schemes *****
m_Diff_opt_response
# Calculates F_i
      a_F[1] = ((a_design_stats[1,1]-
a_design_stats[(m_designs+1)/2,1])*(a_design_stats[1,1]-
a_design_stats[m_designs,1]))**2
      a_F[2] = ((a_design_stats[(m_designs+1)/2,1]-
a_design_stats[1,1])*(a_design_stats[(m_designs+1)/2,1]-
a_design_stats[m_designs,1]))**2
      a_F[3] = ((a_design_stats[m_designs,1]-
a_design_stats[1,1])*(a_design_stats[m_designs,1]-
a_design_stats[(m_designs+1)/2,1]))**2

      INT IRUNSLOOP, ICOUNTRUNS
      IRUNSLOOP = 1
      ICOUNTRUNS = 0
      While IRUNSLOOP < (m_designs+1) DO
      BEGIN

#    Calculates the delta term
      a_design_stats[IRUNSLOOP,12] =
a_design_stats[IRUNSLOOP,11]-a_design_stats[v_lowest,11]

      a_design_stats[IRUNSLOOP,18] =
sqrt((a_design_stats[IRUNSLOOP,1]-a_design_stats[v_lowest,1])**2)

      If v_lowest = 1 Then
      Begin
      a_design_stats[IRUNSLOOP,13] = 1
      End
      Else
      Begin
      a_design_stats[IRUNSLOOP,13] =
(a_design_stats[(m_designs+1)/2,1]-
a_design_stats[IRUNSLOOP,1])*(a_design_stats[m_designs,1]-
a_design_stats[IRUNSLOOP,1])

```

```

        a_design_stats[IRUNSLOOP,13] =
a_design_stats[IRUNSLOOP,13]-(a_design_stats[(m_designs+1)/2,1]-
a_design_stats[v_lowest,1])*(a_design_stats[m_designs,1]-a_design_stats[v_lowest,1])
        a_design_stats[IRUNSLOOP,13] =
(a_design_stats[IRUNSLOOP,13]**2)/a_F[1]
        End

        If v_lowest = (m_designs+1)/2 Then
        Begin
            a_design_stats[IRUNSLOOP,14] = 1
        End
        Else
        Begin
            a_design_stats[IRUNSLOOP,14] = (a_design_stats[1,1]-
a_design_stats[IRUNSLOOP,1])*(a_design_stats[m_designs,1]-
a_design_stats[IRUNSLOOP,1])
            a_design_stats[IRUNSLOOP,14] =
a_design_stats[IRUNSLOOP,14]-(a_design_stats[1,1]-
a_design_stats[v_lowest,1])*(a_design_stats[m_designs,1]-a_design_stats[v_lowest,1])
            a_design_stats[IRUNSLOOP,14] =
(a_design_stats[IRUNSLOOP,14]**2)/a_F[2]
        End

        If v_lowest = m_designs Then
        Begin
            a_design_stats[IRUNSLOOP,15] = 1
        End
        Else
        Begin
            a_design_stats[IRUNSLOOP,15] = (a_design_stats[1,1]-
a_design_stats[IRUNSLOOP,1])*(a_design_stats[(m_designs+1)/2,1]-
a_design_stats[IRUNSLOOP,1])
            a_design_stats[IRUNSLOOP,15] =
a_design_stats[IRUNSLOOP,15]-(a_design_stats[1,1]-
a_design_stats[v_lowest,1])*(a_design_stats[(m_designs+1)/2,1]-
a_design_stats[v_lowest,1])
            a_design_stats[IRUNSLOOP,15] =
(a_design_stats[IRUNSLOOP,15]**2)/a_F[3]
        End

        ICOUNTRUNS = ICOUNTRUNS + a_RunCount[irunsloop,2]
        INC IRUNSLOOP
    END

```

```

# Calculates current alpha_i
a_alpha[1] = a_RunCount[1,2] / Real(ICOUNTRUNS)
a_alpha[2] = a_RunCount[(m_designs+1)/2,2] / Real(ICOUNTRUNS)
a_alpha[3] = a_RunCount[m_designs,2] / Real(ICOUNTRUNS)

IRUNSLOOP = 1
While IRUNSLOOP < (m_designs+1) DO
BEGIN

    a_design_stats[IRUNSLOOP,16]=
(a_design_stats[IRUNSLOOP,13]/a_alpha[1]+a_design_stats[IRUNSLOOP,14]/a_alpha[
2]+a_design_stats[IRUNSLOOP,15]/a_alpha[3])/ICOUNTRUNS

    INC IRUNSLOOP
END

#diff-opt code
# This block calculates sqrt(A1), sqrt(A2), and sqrt(A3) (without T and the variance
included) for the c-opt case

Real Second
Int v_second

If ((v_lowest > 1) and (v_lowest < m_designs)) Then
Begin
If (a_design_stats[(v_lowest-1),12]/sqrt(a_design_stats[(v_lowest-
1),16])) > (a_design_stats[(v_lowest+1),12]/sqrt(a_design_stats[(v_lowest+1),16])) Then
Begin
Second = a_design_stats[(v_lowest+1),1]
v_second = v_lowest+1
End
Else
Begin
Second = a_design_stats[(v_lowest-1),1]
v_second = v_lowest-1
End
End

If v_lowest = m_designs Then
Begin
If (a_design_stats[1,12]/sqrt(a_design_stats[1,16])) >
(a_design_stats[(m_designs-1),12]/sqrt(a_design_stats[(m_designs-1),16])) Then

```



```

        Begin
            Second = a_design_stats[(m_designs-1),1]
            v_second = m_designs-1
        End
    Else
        Begin
            Second = a_design_stats[1,1]
            v_second = 1
        End
    End

    If v_lowest = 1 Then
        Begin
            If
(a_design_stats[m_designs,12]/sqrt(a_design_stats[m_designs,16])) >
(a_design_stats[2,12]/sqrt(a_design_stats[2,16])) Then
                Begin
                    Second = a_design_stats[2,1]
                    v_second = 2
                End
            Else
                Begin
                    Second = a_design_stats[m_designs,1]
                    v_second = m_designs
                End
            End
        End

        v_support = (m_designs+1)/2

        If ((3*a_design_stats[1,1]+a_design_stats[m_designs,1])/4) <
((a_design_stats[v_lowest,1]+Second)/2) Then
            Begin
                If ((a_design_stats[1,1]+a_design_stats[m_designs,1])/2) >
((a_design_stats[v_lowest,1]+Second)/2) Then
                    Begin
                        v_support = v_lowest + v_second -1
                    End
                End
            End

            If ((a_design_stats[1,1]+3*a_design_stats[m_designs,1])/4) >
((a_design_stats[v_lowest,1]+Second)/2) Then
                Begin
                    If ((a_design_stats[1,1]+a_design_stats[m_designs,1])/2) <
((a_design_stats[v_lowest,1]+Second)/2) Then

```

```

        Begin
            v_support = v_lowest + v_second - m_designs
        End
    End

# This line makes it opt alloc to d-opt support points
#v_support = (m_designs+1)/2

    a_Resstats[24] = (a_design_stats[v_support,1]-
a_design_stats[v_lowest,1])*(a_design_stats[m_designs,1]-a_design_stats[v_lowest,1])
    a_Resstats[24] = a_Resstats[24]-(a_design_stats[v_support,1]-
Second)*(a_design_stats[m_designs,1]-Second)
    a_Resstats[24] = a_Resstats[24]/((a_design_stats[1,1]-
a_design_stats[v_support,1])*(a_design_stats[1,1]-a_design_stats[m_designs,1]))

    a_Resstats[25] = (a_design_stats[1,1]-
a_design_stats[v_lowest,1])*(a_design_stats[m_designs,1]-a_design_stats[v_lowest,1])
    a_Resstats[25] = a_Resstats[25]-(a_design_stats[1,1]-
Second)*(a_design_stats[m_designs,1]-Second)
    a_Resstats[25] = a_Resstats[25]/((a_design_stats[v_support,1]-
a_design_stats[1,1])*(a_design_stats[v_support,1]-a_design_stats[m_designs,1]))

    a_Resstats[26] = (a_design_stats[1,1]-
a_design_stats[v_lowest,1])*(a_design_stats[v_support,1]-a_design_stats[v_lowest,1])
    a_Resstats[26] = a_Resstats[26]-(a_design_stats[1,1]-
Second)*(a_design_stats[v_support,1]-Second)
    a_Resstats[26] = a_Resstats[26]/((a_design_stats[m_designs,1]-
a_design_stats[1,1])*(a_design_stats[m_designs,1]-a_design_stats[v_support,1]))

    a_Resstats[24] = sqrt(a_Resstats[24]**2)
    a_Resstats[25] = sqrt(a_Resstats[25]**2)
    a_Resstats[26] = sqrt(a_Resstats[26]**2)

# This block calculates alpha 1, alpha 2, and alpha 3 for x-optimality3

    #v_b = a_Resstats[7]
    #v_B2 = a_ResStats[4]

    If ((a_Resstats[26]<a_Resstats[24]) and
(a_Resstats[26]<a_Resstats[25])) Then
        Begin
            a_Resstats[27] = Round ((m_delta) / (1+
a_Resstats[25]/a_Resstats[24] + a_Resstats[26]/a_Resstats[24]))

```

```

If a_Resstats[27] < 0 Then
    Begin
        a_Resstats[27] = 0
    End

If a_Resstats[27] > m_delta Then
    Begin
        a_Resstats[27] = m_delta
    End

a_Resstats[28] = Round ((m_delta) / (1+
a_Resstats[24]/a_Resstats[25] + a_Resstats[26]/a_Resstats[25]))

If a_Resstats[28] < 0 Then
    Begin
        a_Resstats[28] = 0
    End

If (a_Resstats[27]+a_Resstats[28]) > m_delta Then
    Begin
        a_Resstats[28] = m_delta - a_Resstats[27]
    End

If (a_Resstats[27]+a_Resstats[28]) < m_delta Then
    Begin
        a_Resstats[29] = m_delta - a_Resstats[27] - a_Resstats[28]
    End
ELSE
    Begin
        a_Resstats[29] = 0
    End

End

If ((a_Resstats[24]<a_Resstats[25]) and
(a_Resstats[24]<a_Resstats[26])) Then
    Begin
        a_Resstats[29] = Round ((m_delta) / (1+
a_Resstats[25]/a_Resstats[26] + a_Resstats[24]/a_Resstats[26]))

If a_Resstats[29] < 0 Then
    Begin
        a_Resstats[29] = 0

```

```

End

If a_Resstats[29] > m_delta Then
Begin
a_Resstats[29] = m_delta
End

a_Resstats[28] = Round ((m_delta) / (1+
a_Resstats[24]/a_Resstats[25] + a_Resstats[26]/a_Resstats[25]))

If a_Resstats[28] < 0 Then
Begin
a_Resstats[28] = 0
End

If (a_Resstats[29]+a_Resstats[28]) > m_delta Then
Begin
a_Resstats[28] = m_delta - a_Resstats[29]
End

If (a_Resstats[29]+a_Resstats[28]) < m_delta Then
Begin
a_Resstats[27] = m_delta - a_Resstats[29] - a_Resstats[28]
End
ELSE
Begin
a_Resstats[27] = 0
End

End

If ((a_Resstats[25]<a_Resstats[24]) and
(a_Resstats[25]<a_Resstats[26])) Then
Begin
a_Resstats[27] = Round ((m_delta) / (1+
a_Resstats[25]/a_Resstats[24] + a_Resstats[26]/a_Resstats[24]))

If a_Resstats[27] < 0 Then
Begin
a_Resstats[27] = 0
End

If a_Resstats[27] > m_delta Then

```

```

Begin
  a_Resstats[27] = m_delta
End

a_Resstats[29] = Round ((m_delta) / (1+
a_Resstats[24]/a_Resstats[26] + a_Resstats[25]/a_Resstats[26]))

If a_Resstats[29] < 0 Then
  Begin
    a_Resstats[29] = 0
  End

If (a_Resstats[27]+a_Resstats[29]) > m_delta Then
  Begin
    a_Resstats[29] = m_delta - a_Resstats[27]
  End

If (a_Resstats[27]+a_Resstats[29]) < m_delta Then
  Begin
    a_Resstats[28] = m_delta - a_Resstats[27] - a_Resstats[29]
  End
ELSE
  Begin
    a_Resstats[28] = 0
  End

End

m_pcs_calculation
# This block calculates how far off the center of the best partition is from
# the actual optimal solution

# Did we pick the right solution? (yes or no)
IF V_LOWEST = V_BEST THEN
  BEGIN
    a_results[a_runtotal,1] = a_results[a_runtotal,1] + 1
  END

# x* criteria: How far off are we from the right solution? (difference between the best x
and what we think is the best x)
  a_results[a_runtotal,2] = a_results[a_runtotal,2] + a_Resstats[10]
  a_results[a_runtotal,3] = a_results[a_runtotal,3] +
a_Resstats[10]**2

```

```
# y_hat criteria: How bad is our solution compared to the response at the actual solution?
(difference between y at the best x and y at what we think is the best x)
```

```
    a_results[a_runtotal,4] = a_results[a_runtotal,4] + a_Resstats[9]
    a_results[a_runtotal,5] = a_results[a_runtotal,5] + a_Resstats[9]**2
```

```
# f(x*) criteria: How bad is our solution against the true solution?
```

```
    a_results[a_runtotal,6] = a_results[a_runtotal,6] + (a_Resstats[10]-
v_SOLUTION)**2
```

```
    a_results[a_runtotal,7] = a_results[a_runtotal,7] + (a_Resstats[10]-
v_SOLUTION)**4
```

```
    Allocation_Calculations
```

```
*****
```

```
    m_D_opt
```

```
# D-Opt
```

```
    a_RunCount[1,1] = a_RunCount[1,1] + m_d_opt_runs
```

```
    a_RunCount[(m_designs+1)/2,1] = a_RunCount[(m_designs+1)/2,1]
```

```
+ m_d_opt_runs
```

```
    a_RunCount[m_designs,1] = a_RunCount[m_designs,1] +
```

```
m_d_opt_runs
```

```
    transient_PCS_opt
```

```
    a_RunCount[1,1] = a_RunCount[1,1] + a_Resstats[27]
```

```
    a_RunCount[v_support,1] = a_RunCount[v_support,1] +
```

```
a_Resstats[28]
```

```
    a_RunCount[m_designs,1] = a_RunCount[m_designs,1] +
```

```
a_Resstats[29]
```

```
*****
```

```
*
```

```
External Files
```

```
*
```

```
*****
```

```
ID      Type      File Name  Prompt
```

```
-----
```

```
(null)      a_Results.xls
```

```
(null)      a_scatter.xls
```

## APPENDIX D PROOFS OF LEMMA 4 AND LEMMA 5

**Lemma 4:** When  $P_{hM} = P\{-\tilde{\delta}(x_{hi}) \geq 0\}$  and  $h \neq B$ , the within partition comparisons are determined by the c-optimality criterion where  $x_{hi}$  is selected as one of the three support points and  $\alpha_{hi} = 1.0$ .

Proof: When  $P_{hM} = P\{-\tilde{\delta}(x_{hi}) \geq 0\}$  and  $h \neq B$ ,

$$\frac{\partial U}{\partial \alpha_{hj}} = \phi\left(\frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}}\right) \frac{\hat{\delta}(x_{hi})}{2(\xi_{hi})^{3/2}} \frac{\sigma_h^2}{N_{h\bullet}} \frac{E_{hi,j}^2}{\alpha_{hj}^2} - N_{h\bullet} \lambda$$

Setting  $\frac{\partial U}{\partial \alpha_{hj}} = 0$ , we obtain

$$\phi\left(\frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}}\right) \frac{\hat{\delta}(x_{hi})}{2(\xi_{hi})^{3/2}} \frac{\sigma_h^2}{(N_{h\bullet})^2} \frac{E_{hi,j}^2}{\alpha_{hj}^2} = \lambda \quad (D1)$$

Therefore,

$$\frac{E_{hi,1}^2}{\alpha_{h1}^2} = \frac{E_{hi,s}^2}{\alpha_{hs}^2} = \frac{E_{hi,k}^2}{\alpha_{hk}^2} = \frac{2\lambda (N_{h\bullet})^2 (\xi_{hi})^{3/2}}{\sigma_h^2 \hat{\delta}(x_{hi}) \phi\left(\frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}}\right)}$$

Given a property of the Lagrange polynomial coefficients where  $E_{hi,1} + E_{hi,s} + E_{hi,k} = 1$

(Burden and Faires, 1993), we know that  $E_{hi,j}^2 \neq 0$  for at least one of the support points.

Using the fact that  $\alpha_{h1} + \alpha_{hs} + \alpha_{hk} = 1$  and, assuming for example that  $E_{hi,1}^2 \neq 0$ , we

obtain the result that

$$\alpha_{h1} + \frac{|E_{hi,s}|}{|E_{hi,1}|} \alpha_{h1} + \frac{|E_{hi,k}|}{|E_{hi,1}|} \alpha_{h1} = 1$$

By symmetry of solution, we obtain the general result that

$$\alpha_{hj} = \frac{|E_{hi,j}|}{|E_{hi,1}| + |E_{hi,s}| + |E_{hi,k}|}. \quad (\text{D2})$$

The same results are obtained for the lower bound of the APCS also. For the optimal support point location, we must consider three cases.

Case D-I:  $x_{hi} = x_{h1}$ . For this case, we obtain that  $E_{hi,1} = 1$ ,  $E_{hi,s} = 0$ , and  $E_{hi,k} = 0$ .

Substituting these results into (D2), we obtain  $\alpha_{h1} = 1.0$ .

Case D-II:  $x_{hi} = x_{hk}$ . For this case, we obtain that  $E_{hi,1} = 0$ ,  $E_{hi,s} = 0$ , and  $E_{hi,k} = 1$

resulting in  $\alpha_{hk} = 1.0$ .

Case D-III:  $x_{hi} \neq x_{h1}$  and  $x_{hi} \neq x_{hk}$ . From (59), we know that

$$E_{hi,1} = \left\{ \frac{(x_{hs} - x_{hi})(x_{hk} - x_{hi})}{(x_{h1} - x_{hs})(x_{h1} - x_{hk})} \right\}, \quad E_{hi,s} = \left\{ \frac{(x_{h1} - x_{hi})(x_{hk} - x_{hi})}{(x_{hs} - x_{h1})(x_{hs} - x_{hk})} \right\}, \text{ and}$$

$$E_{hi,k} = \left\{ \frac{(x_{h1} - x_{hi})(x_{hs} - x_{hi})}{(x_{hk} - x_{h1})(x_{hk} - x_{hs})} \right\}.$$



When  $x_{hs} = x_{hi}$ , we obtain that  $E_{hi,1} = 0$ ,  $E_{hi,s} = 1$ , and  $E_{hi,k} = 0$ . Substituting these results into (D2), we obtain  $\alpha_{hs} = 1.0$ . In order to show that  $x_{hs} = x_{hi}$  is an optimal selection of  $x_{hs}$ , we can use the chain rule to establish that

$$\frac{\partial U}{\partial x_{hs}} = - \frac{\partial U}{\partial \xi_{hi}} \frac{\partial \xi_{hi}}{\partial x_{hs}}.$$

Substituting  $\frac{\partial U}{\partial \xi_{hi}}$ , we obtain

$$\frac{\partial U}{\partial x_{hs}} = - \phi\left(\frac{\hat{\delta}(x_{hi})}{\sqrt{\xi_{hi}}}\right) \frac{\hat{\delta}(x_{hi})}{2(\xi_{hi})^{3/2}} \frac{\partial \xi_{hi}}{\partial x_{hs}}.$$

Substituting (D2) into (59), we obtain

$$\xi_{hi} = \frac{\sigma_h^2}{N_{h\bullet}} \left[ |E_{hi,1}| + |E_{hi,s}| + |E_{hi,k}| \right]^2 + \frac{\sigma_B^2}{N_{B\bullet}} \left[ \frac{E_{Bi,1}^2}{\alpha_{B1}} + \frac{E_{Bi,s}^2}{\alpha_{Bs}} + \frac{E_{Bi,k}^2}{\alpha_{Bk}} \right] \quad (D3)$$

Using again the property where  $E_{hi,1} + E_{hi,s} + E_{hi,k} = 1$  (Burden and Faires, 1993), we

know that  $|E_{hi,1}| + |E_{hi,s}| + |E_{hi,k}| \geq 1$ . Thus, when  $x_{hs} > x_{hi}$ ,  $\frac{\partial \xi_{hi}}{\partial x_{hs}} \geq 0$  we obtain that

$$\frac{\partial U}{\partial x_{hs}} \leq 0. \text{ Similarly, when } x_{hs} < x_{hi}, \frac{\partial \xi_{hi}}{\partial x_{hs}} \leq 0 \text{ we obtain that } \frac{\partial U}{\partial x_{hs}} \geq 0.$$

**Lemma 5:** When  $P_{BM} < P_{\Omega M}$ , the within partition comparisons for the best partition are determined by the c-optimality criterion where  $x_{Bb}$  is selected as one of the three support

points and  $\alpha_{Bb} = 1.0$  (for future allocations after initial runs so that we do not have a singular solution).

Proof: When  $P_{BM} < P_{\Omega M}$ ,

$$\frac{\partial U}{\partial \alpha_{Bj}} = \frac{\sigma_B^2}{N_{B\bullet}} \frac{E_{Bb,j}^2}{\alpha_{Bj}^2} \left( \sum_{h \notin \psi} \phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} + \sum_{h \in \psi, M=b} \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{2(\xi_{hb})^{3/2}} \right) - N_{B\bullet} \lambda.$$

Setting  $\frac{\partial U}{\partial \alpha_{Bj}} = 0$ , we obtain

$$\frac{E_{Bb,j}^2}{\alpha_{Bj}^2} = \frac{(N_{B\bullet})^2 \lambda}{\sigma_B^2 \left( \sum_{h \notin \psi} \phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} + \sum_{h \in \psi, M=b} \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{2(\xi_{hb})^{3/2}} \right)}$$

The rest of the proof follows the proof from Lemma 4.

## APPENDIX E PROOFS OF LEMMA 6 AND LEMMA 7

**Lemma 6:** When  $P_{\text{BM}} < P_{\Omega M}$ , the between partition allocations for  $i \neq B$  and  $j \neq B$  are

obtained by  $\frac{N_{i\bullet}}{N_{j\bullet}} = \frac{R_j}{R_i}$ .

Proof: We consider three cases.

Case E-I:

When  $P_{hM} = P\{-\tilde{d}(x_{hA}) \geq 0\}$  or when  $P_{hM} = P\{-\tilde{d}(x_{hZ}) \geq 0\}$  and  $h \neq B$  for both comparisons,

$$\frac{\partial U}{\partial N_{h\bullet}} = \phi\left(\frac{\hat{d}(x_{hM})}{\sqrt{\varsigma_{hM}}}\right) \frac{\hat{d}(x_{hM})}{2(\varsigma_{hM})^{3/2}} \frac{\sigma_h^2}{(N_{h\bullet})^2} \left[ \frac{D_{hM,1}^2}{\alpha_{h1}} + \frac{D_{hM,s}^2}{\alpha_{hs}} + \frac{D_{hM,k}^2}{\alpha_{hk}} \right] - \lambda.$$

Setting  $\frac{\partial U}{\partial N_{h\bullet}} = 0$ , we obtain

$$\phi\left(\frac{\hat{d}(x_{hM})}{\sqrt{\varsigma_{hM}}}\right) \frac{\hat{d}(x_{hM})}{2(\varsigma_{hM})^{3/2}} \frac{\sigma_h^2}{(N_{h\bullet})^2} \left[ \frac{D_{hM,1}^2}{\alpha_{h1}} + \frac{D_{hM,s}^2}{\alpha_{hs}} + \frac{D_{hM,k}^2}{\alpha_{hk}} \right] = \lambda$$

Such that

$$\begin{aligned} & \phi\left(\frac{\hat{d}(x_{iM})}{\sqrt{\varsigma_{iM}}}\right) \frac{\hat{d}(x_{iM})}{2(\varsigma_{iM})^{3/2}} \frac{\sigma_i^2}{(N_{i\bullet})^2} \left[ \frac{D_{iM,1}^2}{\alpha_{i1}} + \frac{D_{iM,s}^2}{\alpha_{is}} + \frac{D_{iM,k}^2}{\alpha_{ik}} \right] = \\ & \phi\left(\frac{\hat{d}(x_{jM})}{\sqrt{\varsigma_{jM}}}\right) \frac{\hat{d}(x_{jM})}{2(\varsigma_{jM})^{3/2}} \frac{\sigma_j^2}{(N_{j\bullet})^2} \left[ \frac{D_{jM,1}^2}{\alpha_{j1}} + \frac{D_{jM,s}^2}{\alpha_{js}} + \frac{D_{jM,k}^2}{\alpha_{jk}} \right] \end{aligned}$$

Chen et al. (2000) provide using an asymptotic allocation rule for where  $T \rightarrow \infty$  and

Glynn and Juneja (2004) provide using a large deviation approach that

$$\left[ \frac{\hat{d}(x_{iM})}{\sqrt{\varsigma_{iM}}} \right]^2 = \left[ \frac{\hat{d}(x_{jM})}{\sqrt{\varsigma_{jM}}} \right]^2.$$

Substituting the results from Chapter 3, we know that

$$\varsigma_{hi} = \frac{\sigma_h^2 \left[ |D_{hM,1}| + |D_{hM,s}| + |D_{hM,k}| \right]^2}{N_{h\bullet}}$$

Therefore,

$$\frac{\left( \hat{d}(x_{iM}) \right)^2}{\frac{\sigma_i^2 \left[ |D_{iM,1}| + |D_{iM,s}| + |D_{iM,k}| \right]^2}{N_{i\bullet}}} = \frac{\left( \hat{d}(x_{jM}) \right)^2}{\frac{\sigma_j^2 \left[ |D_{jM,1}| + |D_{jM,s}| + |D_{jM,k}| \right]^2}{N_{j\bullet}}}$$

$$\frac{N_{i\bullet}}{N_{j\bullet}} = \frac{\sigma_i^2 \left[ |D_{iM,1}| + |D_{iM,s}| + |D_{iM,k}| \right]^2}{\sigma_j^2 \left[ |D_{jM,1}| + |D_{jM,s}| + |D_{jM,k}| \right]^2} \frac{\left( \hat{d}(x_{jM}) \right)^2}{\left( \hat{d}(x_{iM}) \right)^2}$$

Note that these results are very similar to the OCBA results with the major difference being that the Lagrange coefficients serve as an efficiency factor for the within partition allocations.

#### Case E-II:

When  $P_{hM} = P\{-\tilde{\delta}(x_{hM}) \geq 0\}$  and  $h \neq B$  for both comparisons,

$$\frac{\partial U}{\partial N_{h\bullet}} = \phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} \frac{\sigma_h^2}{(N_{h\bullet})^2} \left[ \frac{E_{hM,1}^2}{\alpha_{h1}} + \frac{E_{hM,s}^2}{\alpha_{hs}} + \frac{E_{hM,k}^2}{\alpha_{hk}} \right] - \lambda.$$

Substituting the results from Lemma 4,

$$\xi_{hb} = \frac{\sigma_h^2}{N_{h\bullet}} + \frac{\sigma_B^2}{N_{B\bullet}} \left[ \frac{E_{Bb,l}^2}{\alpha_{B1}} + \frac{E_{Bb,s}^2}{\alpha_{Bs}} + \frac{E_{Bb,k}^2}{\alpha_{Bk}} \right]$$

Using the assumption that  $N_{B\bullet} \gg N_{h\bullet}$  (Chen et al. 2000 and Glynn and Juneja 2004),

$$\xi_{hb} \approx \frac{\sigma_h^2}{N_{h\bullet}}$$

The rest of the proof follows Case E-I such that

$$\frac{N_{i\bullet}}{N_{j\bullet}} = \frac{\sigma_i^2}{\sigma_j^2} \frac{(\hat{\delta}(x_{jb}))^2}{(\hat{\delta}(x_{ib}))^2}$$

Case E-III:

When  $P_{hM} = P\{-\tilde{d}(x_{hA}) \geq 0\}$  or when  $P_{hM} = P\{-\tilde{d}(x_{hZ}) \geq 0\}$  and  $h \neq B$  for one comparison and  $P_{hM} = P\{-\tilde{\delta}(x_{hM}) \geq 0\}$  and  $h \neq B$  for the other comparison,

This case follows from the results for Case E-I and Case E-II such that

$$\frac{N_{i\bullet}}{N_{j\bullet}} = \frac{\sigma_i^2 \left[ |D_{iM,l}| + |D_{iM,s}| + |D_{iM,k}| \right]^2}{\sigma_j^2} \frac{(\hat{\delta}(x_{jb}))^2}{(\hat{d}(x_{iM}))^2}$$

**Lemma 7:** When  $P_{BM} < P_{\Omega M}$ , the between partition allocations for  $h = B$  are obtained by

$$\frac{(N_{B\bullet})^2}{\sigma_B^2} = \sum_{h \in \psi, M=b}^m \frac{(N_{h\bullet})^2}{\sigma_h^2} + \sum_{h \notin \psi}^m \frac{(N_{h\bullet})^2}{\sigma_h^2}.$$

Proof: This proof closely follows the one provided in Chen et al (2000).

$$\frac{\partial U}{\partial N_{B\bullet}} = \frac{\sigma_B^2}{(N_{B\bullet})^2} \left[ \frac{E_{Bb,1}^2}{\alpha_{B1}} + \frac{E_{Bb,s}^2}{\alpha_{Bs}} + \frac{E_{Bb,k}^2}{\alpha_{Bk}} \right] \left( \sum_{h \neq \psi} \phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} + \sum_{h \in \psi, M=b} \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{2(\xi_{hb})^{3/2}} \right) - \lambda$$

Setting  $\frac{\partial U}{\partial N_{B\bullet}} = 0$ , we obtain

$$\left[ \frac{E_{Bb,1}^2}{\alpha_{B1}} + \frac{E_{Bb,s}^2}{\alpha_{Bs}} + \frac{E_{Bb,k}^2}{\alpha_{Bk}} \right] \left( \sum_{h \neq \psi} \phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} + \sum_{h \in \psi, M=b} \phi\left(\frac{\hat{\delta}(x_{hb})}{\sqrt{\xi_{hb}}}\right) \frac{\hat{\delta}(x_{hb})}{2(\xi_{hb})^{3/2}} \right) = \frac{(N_{B\bullet})^2}{\sigma_B^2} \lambda$$

Using the results from Lemma 4 and (D1), it can be shown that for  $h \neq B$

$$\phi\left(\frac{\hat{\delta}(x_{hM})}{\sqrt{\xi_{hM}}}\right) \frac{\hat{\delta}(x_{hM})}{2(\xi_{hM})^{3/2}} = \lambda \frac{(N_{h\bullet})^2}{\sigma_h^2}.$$

Using the results from Lemma 5, we know that  $\left[ \frac{E_{Bb,1}^2}{\alpha_{B1}} + \frac{E_{Bb,s}^2}{\alpha_{Bs}} + \frac{E_{Bb,k}^2}{\alpha_{Bk}} \right] = 1$ .

Substituting these results, we obtain

$$\frac{(N_{B\bullet})^2}{\sigma_B^2} = \sum_{h \neq \psi} \frac{(N_{h\bullet})^2}{\sigma_h^2} + \sum_{h \in \psi, M=b} \frac{(N_{h\bullet})^2}{\sigma_h^2}$$

## APPENDIX F PROOF OF PROPOSITION 1

Intuitively, since we are only sampling on three support points, we should be able to expand the OSD method developed in Chapter 3 to a continuous domain by assuming that the number of design locations goes to infinity. The following proposition establishes a preliminary connection between our results and those from a derivation from the DOE community. However, we have to do some different treatments on the definition of  $PCS$  because the current one will go to zero as the number of design locations goes to infinity.

**Proposition 1:** Assuming that  $PCS > 0$  as  $k \rightarrow \infty$ , for the Interior Design Case presented in (17), using the allocations from (33) and the design locations from (34) results in the same solution obtained by Melas et al. (2003) to find the stationary point of a quadratic equation.

Proof: As the number of designs increases ( $k \rightarrow \infty$ ), the size of the partition between designs gets smaller ( $\Delta \rightarrow 0$ ). For the Interior Design Case, our asymptotic solution compares the best design to an adjacent design. Rewriting (23), the terms assume the form of:

$$D_{M,1} = \left\{ \frac{(x_k - [x_b + \Delta])(x_s - [x_b + \Delta]) - (x_k - x_b)(x_s - x_b)}{(x_1 - x_s)(x_1 - x_k)} \right\}$$

$$D_{M,s} = \left\{ \frac{(x_1 - [x_b + \Delta])(x_k - [x_b + \Delta]) - (x_1 - x_b)(x_k - x_b)}{(x_s - x_1)(x_s - x_k)} \right\}$$

$$D_{M,k} = \left\{ \frac{(x_1 - [x_b + \Delta])(x_s - [x_b + \Delta]) - (x_1 - x_b)(x_s - x_b)}{(x_k - x_1)(x_k - x_s)} \right\}$$

This simplifies to

$$D_{M,1} = \Delta \left\{ \frac{2x_b - x_k - x_s + \Delta}{(x_1 - x_s)(x_1 - x_k)} \right\}$$

$$D_{M,s} = \Delta \left\{ \frac{2x_b - x_1 - x_k + \Delta}{(x_s - x_1)(x_s - x_k)} \right\}$$

$$D_{M,k} = \Delta \left\{ \frac{2x_b - x_1 - x_s + \Delta}{(x_k - x_1)(x_k - x_s)} \right\}$$

By restricting the domain to  $[-1,1]$  as presented by Melas et al. (2003) and by substituting

for  $x_s$  from the first case in (34), we obtain  $x_s = 2x_b + 1$  and that

$$D_{M,1} = \Delta \left\{ \frac{2x_b - 1 - [2x_b + 1] + \Delta}{(-1 - [2x_b + 1])(-1 - 1)} \right\} \text{ or } D_{M,1} = \Delta \left\{ \frac{-2 + \Delta}{2(2x_b + 2)} \right\} = \frac{\Delta}{2} \left\{ \frac{-2 + \Delta}{2x_b + 2} \right\}$$

$$D_{M,s} = \Delta \left\{ \frac{2x_b + \Delta}{(2x_b + 1 - 1)(2x_b + 1 + 1)} \right\} \text{ or } D_{M,s} = \Delta \left\{ \frac{2x_b + \Delta}{2x_b(2x_b + 2)} \right\} = \frac{\Delta}{2} \left\{ \frac{2x_b + \Delta}{x_b(2x_b + 2)} \right\}$$

$$D_{M,k} = \Delta \left\{ \frac{2x_b - [-1] - [2x_b + 1] + \Delta}{(1 - [-1])(1 - [2x_b + 1])} \right\} \text{ or } D_{M,k} = \Delta \left\{ \frac{\Delta}{(2)(2x_b)} \right\} = \frac{\Delta}{2} \left\{ \frac{\Delta}{2x_b} \right\}$$

Using these results,



$$|D_{M,l}| + |D_{M,s}| + |D_{M,k}| = \left| \frac{\Delta}{2} \left\{ \frac{-2 + \Delta}{2x_b + 2} \right\} \right| + \left| \frac{\Delta}{2} \left\{ \frac{2x_b + \Delta}{x_b(2x_b + 2)} \right\} \right| + \left| \frac{\Delta}{2} \left\{ \frac{\Delta}{2x_b} \right\} \right|$$

$$|D_{M,l}| + |D_{M,s}| + |D_{M,k}| = \left| \frac{\Delta}{2} \frac{2x_b}{2x_b} \left\{ \frac{-2 + \Delta}{2x_b + 2} \right\} \right| + \left| \frac{\Delta}{2} \frac{2}{2} \left\{ \frac{2x_b + \Delta}{x_b(2x_b + 2)} \right\} \right| + \left| \frac{\Delta}{2} \frac{(2x_b + 2)}{(2x_b + 2)} \left\{ \frac{\Delta}{2x_b} \right\} \right|$$

$$|D_{M,l}| + |D_{M,s}| + |D_{M,k}| = \frac{\Delta}{4x_b(2x_b + 2)} \{2x_b(2 - \Delta) + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}$$

$$|D_{M,l}| + |D_{M,s}| + |D_{M,k}| = \frac{\Delta}{4x_b(2x_b + 2)} \{4x_b - 2x_b\Delta + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}$$

Substituting these results into (33), we obtain

$$\alpha_1 = \frac{\left| \frac{\Delta}{2} \left\{ \frac{-2 + \Delta}{2x_b + 2} \right\} \right|}{\frac{\Delta}{4x_b(2x_b + 2)} \{4x_b - 2x_b\Delta + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}}$$

or

$$\alpha_1 = \frac{|-4x_b + 2x_b\Delta|}{\{4x_b - 2x_b\Delta + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}}$$

In a similar manner

$$\alpha_s = \frac{|4x_b + 2\Delta|}{\{4x_b - 2x_b\Delta + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}}$$

and

$$\alpha_k = \frac{|2x_b\Delta + 2\Delta|}{\{4x_b - 2x_b\Delta + |4x_b + 2\Delta| + |2x_b\Delta + 2\Delta|\}}$$

As the number of designs increases ( $k \rightarrow \infty$ ) and the size of the partition between designs gets smaller ( $\Delta \rightarrow 0$ ), this results in  $\alpha_1 = 1/2$  and  $\alpha_s = 1/2$ . These are the same

support points and allocations as presented by Melas et al. (2003) for the corresponding case. Similar results are obtained for the second case and third case in (34).

## APPENDIX G PROMODEL CODE FOR PARTITIONING OSD (POSD)

See [www.promodel.com](http://www.promodel.com)

```

*****
*
*
*           Formatted Listing of Model:           *
* C:\Users\Admin\Documents\GMU\Research\Partitioning\Actual POSD code\101127
POSD v14 (Actual code - Cantelli Bounds) x div x.MOD *
*
*****

Time Units:           Minutes
Distance Units:       Feet
*****
*           Locations           *
*****
Name      Cap  Units Stats  Rules  Cost
-----
Experiment_loc 1    1  None  Oldest, ,
Run_Queue    Infinite 1  None  Oldest, ,
Run_Allocation 1    1  None  Oldest, ,
*****
*           Entities           *
*****
Name      Speed (fpm) Stats  Cost
-----
Experiment 150      None
Run      150      None
*****
*           Processing         *
*****

Process           Routing

Entity  Location  Operation  Blk Output  Destination  Rule  Move Logic
-----
Experiment Experiment_loc INT TEMPCOUNT

INC v_iteration

```

```

a_runtotal = 0
v_designs = m_designs

TEMPCOUNT = 1
WHILE TEMPCOUNT < ( (m_partitions)+1) DO
BEGIN
  A_RESSTATS[TEMPCOUNT,1] = 0
  A_RESSTATS[TEMPCOUNT,2] = 0
  A_RESSTATS[TEMPCOUNT,3] = 0
  A_RESSTATS[TEMPCOUNT,4] = 0
  A_RESSTATS[TEMPCOUNT,5] = 0
  A_RESSTATS[TEMPCOUNT,6] = 0
  A_RESSTATS[TEMPCOUNT,7] = 0
  A_RESSTATS[TEMPCOUNT,8] = 0
  A_RESSTATS[TEMPCOUNT,9] = 0
  A_RESSTATS[TEMPCOUNT,10] = 0
  A_RESSTATS[TEMPCOUNT,11] = 0
  A_RESSTATS[TEMPCOUNT,12] = 0
  A_RESSTATS[TEMPCOUNT,13] = 0
  A_RESSTATS[TEMPCOUNT,14] = 0
  A_RESSTATS[TEMPCOUNT,15] = 0
  A_RESSTATS[TEMPCOUNT,16] = 0
  A_RESSTATS[TEMPCOUNT,17] = 0
  A_RESSTATS[TEMPCOUNT,18] = 0
  A_RESSTATS[TEMPCOUNT,19] = 0
  A_RESSTATS[TEMPCOUNT,20] = 0
  A_RESSTATS[TEMPCOUNT,21] = 0
  A_RESSTATS[TEMPCOUNT,22] = 0
  a_Resstats[TEMPCOUNT,23] = (TEMPCOUNT-
1)*m_designs_per+(m_designs_per+1)/2
  INC TEMPCOUNT
END

TEMPCOUNT = 1
WHILE TEMPCOUNT < (m_designs+1) DO
BEGIN
  a_RunCount[TEMPCOUNT,1] = 0
  a_RunCount[TEMPCOUNT,2] = 0

  A_Design_STATS[TEMPCOUNT,1] = 0
  A_Design_STATS[TEMPCOUNT,2] = 0
  A_Design_STATS[TEMPCOUNT,3] = 0
  A_Design_STATS[TEMPCOUNT,4] = 0
  A_Design_STATS[TEMPCOUNT,5] = 0

```

```

        A_Design_STATS[TEMPCOUNT,6] = 0
        A_Design_STATS[TEMPCOUNT,7] = 0
        A_Design_STATS[TEMPCOUNT,8] = 0
        A_Design_STATS[TEMPCOUNT,9] = 0
        A_Design_STATS[TEMPCOUNT,10] = 0

        INC TEMPCOUNT
    END

    Interval = Real(m_Maxx-m_Minx)/Real(m_designs-1)

# This randomly generates optimal solution

# ***** for x^2 and binomial experiment
        # V_SOLUTION = U( (m_Maxx+m_Minx)/2 , (m_Maxx-m_Minx)/2 )
# This fixes the optimal solution
        # V_SOLUTION = 3.0/3.0
        # V_SOLUTION = 5.20338983050847
# for x div x
        V_SOLUTION = 1.01

# This block finds the design with the lowest mean.
    INT bestloop
    REAL distbest, tempint, disttemp

    v_best = 1
    tempint = 0

# ***** for x^2 and binomial experiment
        distbest = (m_minx+tempint-v_solution)*(m_minx+tempint-v_solution)

        bestloop = 2
        WHILE bestloop < (v_designs+1) DO
            BEGIN
                tempint = tempint + interval

# ***** for x^2 and binomial experiment
                disttemp = (m_minx+tempint-v_solution)*(m_minx+tempint-v_solution)

                IF disttemp < distbest Then
                    Begin
                        distbest = disttemp
                        v_best = bestloop
                    End
                End IF
            END
            bestloop = bestloop + 1
        END

```

```

        INC bestloop
    END

```

```

V_EXPERIMENTRUN = 0
v_RUNPER = 0

```

```

TEMPCOUNT = 1
WHILE TEMPCOUNT < (m_increments+1) DO
BEGIN
    INC a_runtotal
    ORDER 1 RUN TO Run_Queue
    Wait 0.4 sec
    INC TEMPCOUNT
END

```

```

                                1 Experiment EXIT      FIRST 1
Run   Run_Queue   Wait 0.1 sec   1 Run   Run_Allocation FIRST 1
Run   Run_Allocation INT TEMPQUEUE, I, J, I2, J2,IOUT

```

```

m_MAIN

```

```

        Wait 1 sec      1 Run   EXIT      FIRST 1

```

```

*****
*                               *
*               Arrivals        *
*****

```

Entity	Location	Qty Each	First Time	Occurrences	Frequency	Logic
Experiment	Experiment_loc 1	0 sec	10000	60 min		

```

*****
*                               *
*               Attributes        *
*****

```

ID	Type	Classification
a_runtotal	Integer	Entity

```

*****
*                               *
*               Variables (global) *
*****

```

ID	Type	Initial value	Stats
ylowest	Real	0	None
xlowest	Real	0	None
v_designs	Integer	m_designs	None
v_SOLUTION	Real	0	None
v_local1	Real	0	None
v_local2	Real	0	None
v_LOWMEAN	Real	0	None
v_LowX	Real	0	None
v_LOWEST	Integer	0	None
v_partition_local_min	Integer	0	None
v_RunPer	Integer	0	None
v_ExperimentRun	Integer	0	None
v_TotalRun	Integer	0	None
V_STAND_VAR	Real	0	None
TempLoop	Integer	0	None
TempRun	Real	0	None
v_Next_C_Run	Integer	0	None
interval	Real	0	None
v_best	Integer	0	None
v_iteration	Integer	0	None
v_varbeta	Real	0	None
v_xc	Real	0	None
v_rho1	Real	0	None
v_rho2	Real	0	None
v_z1	Real	0	None
v_z2	Real	0	None
v_z3	Real	0	None
v_del_bminus	Real	0	None
v_del_bplus	Real	0	None
v_b	Real	0	None
v_B2	Real	0	None
v_B2_actual	Integer	0	None
v_c0	Real	0	None
v_c1	Real	0	None
v_c2	Real	0	None
v_support	Integer	0	None
v_best_partition	Integer	0	None
v_min_signal_noise_value	Real	0	Time Series

#

#Used for OCBA code (determines partition with lowest signal noise other than best partition)

v_Second_Lowest	Integer	0	Time Series
-----------------	---------	---	-------------

```
#Used for OCBA code
v_CoefSum      Real      0      Time Series
#
```

```
#Used for OCBA Code
v_CoefSqr      Real      0      Time Series
```

```
*****
*                      *
*              Arrays              *
*                      *
*****
```

ID	Dimensions	Type	Import File	Export File	Disable	Persist
----	------------	------	-------------	-------------	---------	---------

```
#
#COL 1: Location of X
#COL 2: Sum of X
#COL 3: Sum of X^2
#COL 4: Sum of X^3
#COL 5: Sum of X^4
#COL 6: Sum of y
#COL 7: Sum of yx
#COL 8: Sum of yx^2
#COL 9: Sum of y^2
#COL 10: Variance of runs for this design (Actually x_h*(info matrix)*x_h
- need to multiply by MSE for variance
#COL 11: Predicted response at each location
#COL 12: Delta b,i (the within partition comparison)
#COL 13: Li,1
#COL 14: Li,2
#COL 15: Li,3
#COL 16: Sigma^2_b,i (old - does not use all runs but only estimates using
three point asymptotic points)
#COL 17: Sigma^2_b,i (new - using all runs from info matrix)
#COL 18: delta b,B (the between partition comparison)
#COL 19: Sigma between global and all others (need to adjust this so it
is not calculated for the best partition within comparisons)
#COL 20: Signal to noise ratio for design to local best
#COL 21: Signal to noise ratio for design to global best
a_Design_stats 130,21      Real      None      No
a_RunCount     130,2      Integer   None      No
```

```
#
#Each Row is a partition
#
#COL 1:
#COL 2:
```



#COL 3:  
 #COL 4: beta2 term  
 #COL 5: beta1 term  
 #COL 6: beta0 term  
 #COL 7: Design location with lowest OSD signal to noise (only A and Z -  
 local best info captured in COL 18 and COL 11)  
 #COL 8: Value of lowest OSD signal to noise  
 #COL 9: Design location with lowest FULL signal to noise  
 #COL 10: Value of lowest FULL signal to noise  
 #COL 11: Value of local best signal to noise  
 #COL 12: Type if signal to noise ratio used (OSD=1, Full=2, or Local=3)  
 #COL 13: Value of signal to noise ratio used (min of OSD, Full, or Local)  
 #COL 14: Used for OCBA code (essentially (signal/noise)^2 except for special  
 cases of the best partition)  
 #COL 15:  
 #COL 16: Cantelli sum for Quad Bound  
 #COL 17: Cantelli sum for Full Bound  
 #COL 18: Lowest design for the partition  
 #COL 19: Value of the Lowest design for the partition  
 #COL 20: Second  
 #COL 21: v\_second  
 #COL 22:  $s^2$  (MSE) =  $\sum (y_i - \hat{y}_i)^2$   
 #COL 23: v\_support for the partition  
 #COL 24: A1 (for c-opt)  
 #COL 25: A2  
 #COL 26: A3  
 #COL 27: alpha 1  
 #COL 28: alpha 2  
 #COL 29: alpha 3  
 #COL 30: Runs count for the partition  
 #COL 31: Inverse 1,1  
 #COL 32: Inverse 1,2  
 #COL 33: Inverse 1,3  
 #COL 34: Inverse 2,1  
 #COL 35: Inverse 2,2  
 #COL 36: Inverse 2,3  
 #COL 37: Inverse 3,1  
 #COL 38: Inverse 3,2  
 #COL 39: Inverse 3,3  
 #  
 a\_ResStats    61,39    Real                    None        No  
 #  
 #COL 1: P{CS}  
 #COL 2: Mean Distance from local

```

#COL 3: Squared Distance from local
#COL 4: Mean y_hat(x*)
#COL 5: Squared y_hat(x*)
#COL 6: Mean f(x*)
#COL 7: Squared f(x*)
#COL 8: rho1
#COL 9: rho2
#COL 10: Allocations to Design 1
#COL 20: Allocations to Design 11

```

```

a_Results      1501,130   Real          a_Results.xls None      No
#
#COL 1: Coefficients
#COL 2: Temp Column
a_B            3,3       Real          None      No
a_RHS          3        Real          None      No
a_INV          3,3       Real          None      No
FTF_MATRIX     3,3       Real          None
a_TEMP         6,6       Real          None      No
a_F            3        Real          None      No
a_E            3        Real          None      No
a_alpha        3        Real          None      No
a_function     60,1      Real    f1 function values.xlsx  None      No
a_selected_best 1501,60   Integer          None      No

```

```

*****
*                               *
*                               *
*****

```

```

ID          Text
-----
m_nzero      2
m_initial_loops  2
m_delta      14
m_restricted  1
m_MAIN       REAL RunValue
              INT Tempmesh, INEXTRUNS

```

```

# STEP 1 RUN ALLOCATIONS: This block sets the number of runs for each partition to
the value of nzero (initial iterations) and generates the mesh.

```

```

      IF A_RUNTOTAL < m_initial_loops THEN
      BEGIN

```

```

# Conducts 2 iterations of D-opt instead of runs to entire domain

```

```

        INEXTRUNS = 1
        WHILE INEXTRUNS < (m_partitions+1) DO
            BEGIN
                a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] =
a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] + 2*m_d_opt_runs
                a_RunCount[(INEXTRUNS-
1)*m_designs_per+(m_designs_per+1)/2,1] = a_RunCount[(INEXTRUNS-
1)*m_designs_per+(m_designs_per+1)/2,1] + 2*m_d_opt_runs
                a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] = a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] + 2*m_d_opt_runs
                INC INEXTRUNS
            END

# During initialization, this block generates the location of each design.
        Tempmesh = 1
        While Tempmesh < (v_designs+1) DO
            Begin
                a_Design_stats[Tempmesh,1]= m_Minx + Real(Tempmesh-
1)*Real(m_Maxx-m_Minx)/Real(m_designs-1)
                INC Tempmesh
            End
        END

# This block does the subsequent iterations.

        ELSE
            BEGIN

                INEXTRUNS = 1
                WHILE INEXTRUNS < (m_partitions+1) DO
                    BEGIN

                        If (a_Resstats[v_best_partition,8] < v_min_signal_noise_value) Then
                            Begin
                                If INEXTRUNS = v_best_partition Then
                                    Begin

##### Theorem: Only to Best with OSD
# Just a counter to see how many time we use this rule over time
                                    INC a_results[a_runtotal,3]

                                    v_support = a_Resstats[INEXTRUNS,23]

```

```

        a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] =
a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] + a_Resstats[INEXTRUNS,27]
        a_RunCount[v_support,1] = a_RunCount[v_support,1] +
a_Resstats[INEXTRUNS,28]
        a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] = a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] + a_Resstats[INEXTRUNS,29]
    End
End
Else
Begin

##### Theorem: C-opt to best and rules for others
    If INEXTRUNS = v_best_partition Then
        Begin
# Just a counter to see how many time we use this rule over time
        INC a_results[a_runtotal,4]
        v_support = a_Resstats[INEXTRUNS,18]
        a_RunCount[v_support,1] = a_RunCount[v_support,1] +
a_Resstats[INEXTRUNS,1]
        End
    Else
        Begin
# c-opt to local best
        IF (a_Resstats[INEXTRUNS,11] < a_Resstats[INEXTRUNS,8])
THEN
            Begin
# Just a counter to see how many time we use this rule over time
            INC a_results[a_runtotal,5]
            v_support = a_Resstats[INEXTRUNS,18]
            a_RunCount[v_support,1] = a_RunCount[v_support,1] +
a_Resstats[INEXTRUNS,1]
            End
        Else
            Begin
# c-opt to full (see other set of code at the bottom *****)
#            IF ((a_Resstats[INEXTRUNS,10]) < a_Resstats[INEXTRUNS,8]) THEN
#                #            IF ((m_designs_per/(1+a_Resstats[INEXTRUNS,10]**2)) <
(3/(1+a_Resstats[INEXTRUNS,8]**2))) THEN
#                    IF ((a_Resstats[INEXTRUNS,17]) <
a_Resstats[INEXTRUNS,16]) THEN
#
#                        Begin

```

```

# Just a counter to see how many time we use this rule over time
    INC a_results[a_runtotal,6]
    v_support = a_Resstats[INEXTRUNS,9]
    a_RunCount[v_support,1] = a_RunCount[v_support,1] +
a_Resstats[INEXTRUNS,1]
    End
# OSD
    Else
    Begin
# Just a counter to see how many time we use this rule over time
    INC a_results[a_runtotal,7]
    v_support = a_Resstats[INEXTRUNS,23]
    a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] =
a_RunCount[(INEXTRUNS-1)*m_designs_per+1,1] + a_Resstats[INEXTRUNS,27]
    a_RunCount[v_support,1] = a_RunCount[v_support,1] +
a_Resstats[INEXTRUNS,28]
    a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] = a_RunCount[(INEXTRUNS-
1)*m_designs_per+m_designs_per,1] + a_Resstats[INEXTRUNS,29]
    End
    End

    End
    INC INEXTRUNS
END

END

```

# STEP 2 RUN GENERATION: This block checks to see how many total runs a design has been allocated and how many it currently has.

# If it needs more, it calls the macro for allocating a new run based upon the simulated underlying distribution.

```

Temploop = 1
While Temploop < (v_designs+1) DO
    Begin
        While a_RunCount[Temploop,2] < a_RunCount[Temploop,1] DO
            Begin

                INC V_totalrun
                INC v_ExperimentRun

```

```

                                INC a_results[a_runtotal,Temploop+9]
                                # RunValue = (a_design_stats[Temploop,1]-
v_SOLUTION)**2 + N(0,1)
                                RunValue = 10*a_design_stats[Temploop,1]+
10/a_design_stats[Temploop,1] + N(0,1)
                                # RunValue = (a_design_stats[Temploop,1]-
v_SOLUTION)**4 + N(0,1)
                                # RunValue = a_function[Temploop,1]+ N(0,1)

                                A_Design_STATS[Temploop,2] =
A_Design_STATS[Temploop,2] + a_design_stats[Temploop,1]
                                A_Design_STATS[Temploop,3] =
A_Design_STATS[Temploop,3] + a_design_stats[Temploop,1]**2
                                A_Design_STATS[Temploop,4] =
A_Design_STATS[Temploop,4] + a_design_stats[Temploop,1]**3
                                A_Design_STATS[Temploop,5] =
A_Design_STATS[Temploop,5] + a_design_stats[Temploop,1]**4
                                A_Design_STATS[Temploop,6] =
A_Design_STATS[Temploop,6] + RunValue
                                A_Design_STATS[Temploop,7] =
A_Design_STATS[Temploop,7] + RunValue*a_design_stats[Temploop,1]
                                A_Design_STATS[Temploop,8] =
A_Design_STATS[Temploop,8] + RunValue*a_design_stats[Temploop,1]**2
                                A_Design_STATS[Temploop,9] =
A_Design_STATS[Temploop,9] + RunValue**2
                                INC a_RunCount[Temploop,2]
                                End

                                INC Temploop
                                End

```

# STEP 3 Generates a response surface for each partition (and also uses the loop to initialize the MSE to zero)

```

REAL AMAX, SWITCH, TEMPCOEF, TEMPDIFF
INT NMAX, II, JJ, III, IOUTLOOP
INT IJKL,IMN, IIKK, IIJJ
REAL TOL, TEMPTOL

IIKK = 1
WHILE IIKK < (m_partitions+1) DO
BEGIN

    m_Info_matrix

```

```

a_Resstats[IKK,30] = 0
a_Resstats[IKK,22] = 0

INC IKK
END

# STEP 4: Determines the total runs allocated for the entire partition and the response at
each discrete point
IIJJ = 1
IKK = 1
ylowest =
a_Resstats[IKK,6]+a_Resstats[IKK,5]*a_design_stats[IIJJ,1]+a_Resstats[IKK,4]*a_de
sign_stats[IIJJ,1]**2
v_lowest = 1

WHILE IKK < (m_partitions+1) DO
BEGIN

a_Resstats[IKK,19] =
a_Resstats[IKK,6]+a_Resstats[IKK,5]*a_design_stats[IIJJ,1]+a_Resstats[IKK,4]*a_de
sign_stats[IIJJ,1]**2
a_Resstats[IKK,18] = IIJJ

Int IIJJCount
IIJJCount = 1
WHILE IIJJCount < ( (m_designs_per)+1) DO
BEGIN
# Calculates total runs allocated for the entire partition
a_Resstats[IKK,30] = a_Resstats[IKK,30] +
Real(a_RunCount[IIJJ,2])

# Calculates the response of each design location
a_design_stats[IIJJ,11]=
a_Resstats[IKK,6]+a_Resstats[IKK,5]*a_design_stats[IIJJ,1]+a_Resstats[IKK,4]*a_de
sign_stats[IIJJ,1]**2

#Calculates info matrix portion of the variance of each design location (uses symmetry of
the matrix so it is (1,1)+2*(1,2)x+(2*(1,3)+(2,2))*x^2+2*(2,3)x^3+(3,3)*x^4
a_design_stats[IIJJ,10]=
a_Resstats[IKK,31]+(2*a_Resstats[IKK,32])*a_design_stats[IIJJ,1]
a_design_stats[IIJJ,10]= a_design_stats[IIJJ,10] +
(2*a_Resstats[IKK,33]+a_Resstats[IKK,35])*a_design_stats[IIJJ,1]**2

```

```

a_design_stats[IJJ,10]= a_design_stats[IJJ,10] +
(2*a_Resstats[IKK,36])*a_design_stats[IJJ,1]**3
a_design_stats[IJJ,10]= a_design_stats[IJJ,10] +
(a_Resstats[IKK,39])*a_design_stats[IJJ,1]**4

```

# Calculates the sums for MSE for the partition (each is initialized to zero using the loop in Step 3 above and will be divided by (n-2) after each partitions loop)

```

a_Resstats[IKK,22] = a_Resstats[IKK,22] +
a_design_stats[IJJ,9] - 2*a_design_stats[IJJ,6]*a_design_stats[IJJ,11] +
Real(a_RunCount[IJJ,2])*(a_design_stats[IJJ,11]**2)

```

# checks if global min

```

IF a_design_stats[IJJ,11] < ylowest then
BEGIN
ylowest = a_design_stats[IJJ,11]
v_lowest = IJJ
v_best_partition = IKK
END

```

# checks if min of partition

```

IF a_design_stats[IJJ,11] < a_Resstats[IKK,19] then
BEGIN
a_Resstats[IKK,19] = a_design_stats[IJJ,11]
a_Resstats[IKK,18] = IJJ
END

```

```

INC IJJCount
INC IJJ
END

```

# Divides the sum for the MSE by (n-2)

```

a_Resstats[IKK,22] = a_Resstats[IKK,22]/(a_Resstats[IKK,30]-2)

```

```

INC IKK
END

```

# STEP 4a: Determines the comparison terms for each design

```

IJJ = 1
IKK = 1
v_min_signal_noise_value = 999999
v_second_lowest = 999999
Real second_lowest_value = 999999

```

```

WHILE IKK < (m_partitions+1) DO
BEGIN

```



```

IIJJCount = 1

# Initializes minimum signal to noise ratio terms
a_Resstats[IKK,8] = 99999
a_Resstats[IKK,10] = 99999
a_Resstats[IKK,16] = 0
a_Resstats[IKK,17] = 0

WHILE IIJJCount < ( (m_designs_per)+1) DO
BEGIN

# Delta term with local best
a_design_stats[IIJJ,12]= a_design_stats[IIJJ,11]-
a_design_stats[a_Resstats[IKK,18],11]

# Variance term with local best (uses symmetry of the matrix so it is (2,2))*(x_i-
x_b)^2+2*(2,3)(x_i-x_b)(x_i^2-x_b^2)+(3,3)*(x_i^2-x_b^2)^2
a_design_stats[IIJJ,17]=
(a_Resstats[IKK,35])*(a_design_stats[IIJJ,1]-
a_design_stats[a_Resstats[IKK,18],1])**2
a_design_stats[IIJJ,17]= a_design_stats[IIJJ,17] +
(2*a_Resstats[IKK,36])*(a_design_stats[IIJJ,1]-
a_design_stats[a_Resstats[IKK,18],1])*(a_design_stats[IIJJ,1]**2-
a_design_stats[a_Resstats[IKK,18],1]**2)
a_design_stats[IIJJ,17]= a_design_stats[IIJJ,17] +
(a_Resstats[IKK,39])*((a_design_stats[IIJJ,1]**2-
a_design_stats[a_Resstats[IKK,18],1]**2)**2)
a_design_stats[IIJJ,17]=
a_design_stats[IIJJ,17]*a_Resstats[IKK,22]

# Signal to noise ratio for local comparison

If (a_design_stats[IIJJ,12] > 0) Then
Begin
a_design_stats[IIJJ,20]=
a_design_stats[IIJJ,12]/(sqrt(a_design_stats[IIJJ,17]))
End
Else
Begin
a_design_stats[IIJJ,20]= 0
End

# Delta term with global best

```

```

        a_design_stats[IIJJ,18]= a_design_stats[IIJJ,11]-
a_design_stats[v_lowest,11]

# Variance term with global best
        a_design_stats[IIJJ,19]=
a_design_stats[IIJJ,10]*a_Resstats[IKKK,22]+a_design_stats[v_lowest,10]*a_Resstats[v_
best_partition,22]

# Signal to noise ratio for global (FULL) comparison

        If (a_design_stats[IIJJ,18] > 0) Then
            Begin
                a_design_stats[IIJJ,21]=
a_design_stats[IIJJ,18]/(sqrt(a_design_stats[IIJJ,19]))

                If (a_design_stats[IIJJ,21] <
a_Resstats[IKKK,10]) Then
                    Begin
                        a_Resstats[IKKK,10] = a_design_stats[IIJJ,21]
                        a_Resstats[IKKK,9] = IIJJ
                    End

                    If (a_design_stats[IIJJ,21] <
v_min_signal_noise_value) Then
                        Begin
                            If IKKK = v_best_partition Then
                                Begin
                                    v_min_signal_noise_value = a_design_stats[IIJJ,21]
                                End
                            Else
                                Begin
                                    v_min_signal_noise_value = a_design_stats[IIJJ,21]
                                End
                            End

                        End

                    End

                End

            End
        Else
            Begin
                a_design_stats[IIJJ,21]= 0
            End

# For Cantelli Bound

```

```

                                a_Resstats[IKK,17] = a_Resstats[IKK,17] +
1/(1+a_design_stats[IJJ,21]**2)

                                INC IJJCount
                                INC IJJ
                                END

# Signal to noise ratio value for local best
                                a_Resstats[IKK,11] = a_design_stats[a_Resstats[IKK,18],21]

# Signal to noise ratio for OSD comparion (A and Z)
# Initializes for the middle test
                                a_Resstats[IKK,8] = 999999

# Local best is first design in partition
                                If (a_Resstats[IKK,18]=(IJJ-m_designs_per)) Then
                                    Begin
                                        If (a_design_stats[IJJ+1-
m_designs_per,20] < a_design_stats[IJJ-1,20]) Then
                                            Begin
                                                a_Resstats[IKK,8] = a_design_stats[IJJ+1-
m_designs_per,20]
                                                a_Resstats[IKK,7] = IJJ+1-m_designs_per
                                            End
                                        Else
                                            Begin
                                                a_Resstats[IKK,8] = a_design_stats[IJJ-1,20]
                                                a_Resstats[IKK,7] = IJJ-1
                                            End
                                        End
                                    End

# for Cantelli Quad bound
                                a_Resstats[IKK,16] =
1/(1+a_design_stats[IJJ,21]**2)+1/(1+a_design_stats[IJJ-
1,20]**2)+1/(1+a_design_stats[IJJ+1-m_designs_per,20]**2)

                                End

# Local best is last design in partition
                                If (a_Resstats[IKK,18]=(IJJ-1)) Then
                                    Begin
                                        If (a_design_stats[IJJ-m_designs_per,20] <
a_design_stats[IJJ-2,20]) Then
                                            Begin

```

```

a_Resstats[IKK,8] = a_design_stats[IJJ-
m_designs_per,20]
a_Resstats[IKK,7] = IJJ-m_designs_per
End
Else
Begin
a_Resstats[IKK,8] = a_design_stats[IJJ-2,20]
a_Resstats[IKK,7] = IJJ-2
End

# for Cantelli Quad bound
a_Resstats[IKK,16] =
1/(1+a_design_stats[IJJ,21]**2)+1/(1+a_design_stats[IJJ-
2,20]**2)+1/(1+a_design_stats[IJJ-m_designs_per,20]**2)

End

# Local best is in middle of partition (actually computes all and then the next two blocks
recompute if first or last in partition)
If a_Resstats[IKK,8] = 999999 Then
Begin
If (a_design_stats[a_Resstats[IKK,18]-
1,20] < a_design_stats[a_Resstats[IKK,18]+1,20]) Then
Begin
a_Resstats[IKK,8] = a_design_stats[a_Resstats[IKK,18]-
1,20]
a_Resstats[IKK,7] = a_Resstats[IKK,18]-1
End
Else
Begin
a_Resstats[IKK,8] =
a_design_stats[a_Resstats[IKK,18]+1,20]
a_Resstats[IKK,7] = a_Resstats[IKK,18]+1
End

# for Cantelli Quad bound
a_Resstats[IKK,16] =
1/(1+a_design_stats[a_Resstats[IKK,18],21]**2)+1/(1+a_design_stats[a_Resstats[IKK,
18]-1,20]**2)+1/(1+a_design_stats[a_Resstats[IKK,18]+1,20]**2)

End

# Determines support point within the partition for OSD
m_support_point

```

```

# Determines which signal to noise ratio in each partition is the most dominating
# c-opt to local best
      IF (a_Resstats[IKK,11] < a_Resstats[IKK,8]) THEN
      Begin
        a_Resstats[IKK,12] = 3
        a_Resstats[IKK,13] = a_Resstats[IKK,11]
      End
    Else
      Begin

# c-opt to full (see other set of code at the top *****)
#      IF ((m_designs_per/(1+a_Resstats[IKK,10]**2)) <
(3/(1+a_Resstats[IKK,8]**2))) THEN
        IF ((a_Resstats[IKK,17]) < a_Resstats[IKK,16]) THEN
        Begin
          a_Resstats[IKK,12] = 2
          a_Resstats[IKK,13] = a_Resstats[IKK,10]
        End

# OSD
        Else
        Begin
          a_Resstats[IKK,12] = 1
          a_Resstats[IKK,13] = a_Resstats[IKK,8]
        End
      End

# Determines the partition considered "second best" for the OCBA routine
      IF IKK <> v_best_partition Then
      Begin
        #      If a_Resstats[IKK,12] > 1 Then
        #      Begin
        #        If a_Resstats[IKK,13] < second_lowest_value Then
        #        Begin
        #          second_lowest_value = a_Resstats[IKK,13]
        #          v_second_lowest = IKK
        #        End
        #      End
      End

      INC IKK
    END

# Just put into the code in case none of the partitions are using full or local comparisons

```

```

If v_second_lowest = 999999 Then
  Begin
    If v_best_partition = 1 Then
      Begin
        v_second_lowest = 2
      End
    Else
      Begin
        v_second_lowest = v_best_partition - 1
      End
    End
  End

  If (a_Resstats[v_best_partition,8] < v_min_signal_noise_value) Then
    Begin
      a_Resstats[v_best_partition,1] = m_partitions*m_delta
    End
  Else
    Begin
      m_OCBA
    End
  End

# Deterimines the OSD allocations for each partition
Real Second
Int v_second
IIKK = 1
WHILE IIKK < (m_partitions+1) DO
  BEGIN
    v_partition_local_min = a_Resstats[IIKK,18]
    Second = a_Resstats[IIKK,7]
    v_support = a_Resstats[IIKK,23]
    m_OSD
    INC IIKK
  END

  inc a_selected_best[a_runtotal,v_lowest]

# Did we pick the right solution? (yes or no)
IF V_LOWEST = V_BEST THEN
  BEGIN
    a_results[a_runtotal,1] = a_results[a_runtotal,1] + 1
  END
m_Info_matrix      # This block is used to build the FTF

```

```

II = 1
While II < (NDIM+1) DO
  Begin
    JJ = 1
    a_RHS[II] = 0
    While JJ < (NDIM+1) DO
      Begin
        FTF_MATRIX[JJ,II] = 0
        INC JJ
      END
    END
    INC II
  END

II = 1
WHILE II < ( (m_designs_per)+1) DO
  BEGIN
    FTF_MATRIX[1,1] = FTF_MATRIX[1,1] + a_RunCount[(IIKK-
1)*m_designs_per+II,2]
    FTF_MATRIX[1,2] = FTF_MATRIX[1,2] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,2]
    FTF_MATRIX[1,3] = FTF_MATRIX[1,3] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,3]
    FTF_MATRIX[2,1] = FTF_MATRIX[2,1] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,2]
    FTF_MATRIX[2,2] = FTF_MATRIX[2,2] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,3]
    FTF_MATRIX[2,3] = FTF_MATRIX[2,3] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,4]
    FTF_MATRIX[3,1] = FTF_MATRIX[3,1] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,3]
    FTF_MATRIX[3,2] = FTF_MATRIX[3,2] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,4]
    FTF_MATRIX[3,3] = FTF_MATRIX[3,3] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,5]
    a_RHS[1] = a_RHS[1] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,6]
    a_RHS[2] = a_RHS[2] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,7]
    a_RHS[3] = a_RHS[3] + A_Design_STATS[(IIKK-
1)*m_designs_per+II,8]

    INC II
  END

```

# This block is used to invert the FTF matrix

```
II = 1
While II < (NDIM+1) DO
  Begin
    JJ = 1
    While JJ < (NDIM+1) DO
      Begin
        A_TEMP[JJ,II] = FTF_MATRIX[JJ,II]
        INC JJ
      END
    END
    INC II
  END
```

```
II = 1
While II < (NDIM+1) DO
  Begin
    JJ = (NDIM+1)
    While JJ < (2*NDIM+1) DO
      Begin
        A_TEMP[II,JJ] = 0
        INC JJ
      END
    END
    A_TEMP[II,NDIM+II] = 1
    INC II
  END
```

# This loop row reduces the first N columns of the matrix

```
II = 1
While II < (NDIM+1) DO
  Begin
```

# This loop does maximum element pivoting for each column and then

# divides the row by the first element in the row

```
    AMAX = A_TEMP[II,II]
    NMAX = II

    JJ = II+1
    WHILE JJ < (NDIM+1) DO
      BEGIN
        IF SQRT(A_TEMP[JJ,II]**2) > SQRT(AMAX**2) THEN
          BEGIN
```



```

        AMAX = A_TEMP[JJ,II]
        NMAX = JJ
        END
        INC JJ
    END

    IF II <> NMAX THEN
        BEGIN
            JJ = II
            WHILE JJ < (2*NDIM+1) DO
                BEGIN
                    SWITCH = A_TEMP[II,JJ]
                    A_TEMP[II,JJ] = A_TEMP[NMAX,JJ]/AMAX
                    A_TEMP[NMAX,JJ] = SWITCH
                    INC JJ
                END
            END
        ELSE
            BEGIN
                JJ = 1
                WHILE JJ < (2*NDIM+1) DO
                    BEGIN
                        A_TEMP[II,JJ] = A_TEMP[II,JJ]/AMAX
                        INC JJ
                    END
                END
            END
        END
    END

```

# This loop reduces all other elements in the column to zero

```

        JJ = 1
        WHILE JJ < (NDIM+1) DO
            BEGIN
                IF JJ <> II THEN
                    BEGIN
                        TEMPCOEF = -A_TEMP[JJ,II]
                        III = II
                        WHILE III < (2*NDIM+1) DO
                            BEGIN
                                A_TEMP[JJ,III] = A_TEMP[JJ,III] +
TEMPCOEF*A_TEMP[II,III]
                                INC III
                            END
                        END
                    END
                END
            END
        END
    END

```

```

        END
        INC JJ
    END

```

```

    INC II
END

```

# This loop establishes the inverse matrix using the non-reduced rows

```

    II = 1
    WHILE II < (NDIM+1) DO
    BEGIN
        JJ = (NDIM+1)
        WHILE JJ < (2*NDIM+1) DO
        BEGIN
            A_INV[II,JJ-NDIM] = A_TEMP[II,JJ]
            INC JJ
        END
        INC II
    END
END

```

# This loop establishes the inverse matrix using the non-reduced rows

```

        a_RESstats[IKK,6] = a_INV[1,1]*a_RHS[1] + a_INV[1,2]*a_RHS[2]
+ a_INV[1,3]*a_RHS[3]
        a_RESstats[IKK,5] = a_INV[2,1]*a_RHS[1] + a_INV[2,2]*a_RHS[2]
+ a_INV[2,3]*a_RHS[3]
        a_RESstats[IKK,4] = a_INV[3,1]*a_RHS[1] + a_INV[3,2]*a_RHS[2]
+ a_INV[3,3]*a_RHS[3]

```

```

a_RESstats[IKK,31] = a_INV[1,1]
a_RESstats[IKK,32] = a_INV[1,2]
a_RESstats[IKK,33] = a_INV[1,3]
a_RESstats[IKK,34] = a_INV[2,1]
a_RESstats[IKK,35] = a_INV[2,2]
a_RESstats[IKK,36] = a_INV[2,3]
a_RESstats[IKK,37] = a_INV[3,1]
a_RESstats[IKK,38] = a_INV[3,2]

```

```

        a_RESstats[IKK,39] = a_INV[3,3]
m_support_point    # Determines x_s

v_support = (IKK-1)*m_designs_per+(m_designs_per+1)/2

    If ((3*a_design_stats[(IKK-
1)*m_designs_per+1,1]+a_design_stats[(IKK-1)*m_designs_per+m_designs_per,1])/4)
< ((a_design_stats[a_Resstats[IKK,18],1]+a_design_stats[a_Resstats[IKK,7],1])/2)
Then
        Begin
            If ((a_design_stats[(IKK-
1)*m_designs_per+1,1]+a_design_stats[(IKK-1)*m_designs_per+m_designs_per,1])/2)
> ((a_design_stats[a_Resstats[IKK,18],1]+a_design_stats[a_Resstats[IKK,7],1])/2)
Then
                Begin
                    v_support = (IKK-1)*m_designs_per+(a_Resstats[IKK,18]-(IKK-
1)*m_designs_per) + (a_Resstats[IKK,7]-(IKK-1)*m_designs_per) -1
                End
            End
        End

        If ((a_design_stats[(IKK-
1)*m_designs_per+1,1]+3*a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1])/4) >
((a_design_stats[a_Resstats[IKK,18],1]+a_design_stats[a_Resstats[IKK,7],1])/2) Then
            Begin
                If ((a_design_stats[(IKK-
1)*m_designs_per+1,1]+a_design_stats[(IKK-1)*m_designs_per+m_designs_per,1])/2)
< ((a_design_stats[a_Resstats[IKK,18],1]+a_design_stats[a_Resstats[IKK,7],1])/2)
Then
                    Begin
                        v_support = (IKK-1)*m_designs_per+(a_Resstats[IKK,18]-(IKK-
1)*m_designs_per) + (a_Resstats[IKK,7]-(IKK-1)*m_designs_per) - m_designs_per
                    End
                End
            End

        a_RESstats[IKK,23] = v_support
m_OSD    # Calculates the Q terms
        a_Resstats[IKK,24] = (a_design_stats[v_support,1]-
a_design_stats[v_partition_local_min,1])*(a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]-a_design_stats[v_partition_local_min,1])
        a_Resstats[IKK,24] = a_Resstats[IKK,24]-
(a_design_stats[v_support,1]-Second)*(a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]-Second)

```

```

a_Resstats[IKK,24] = a_Resstats[IKK,24]/((a_design_stats[(IKK-
1)*m_designs_per+1,1]-a_design_stats[v_support,1])*(a_design_stats[(IKK-
1)*m_designs_per+1,1]-a_design_stats[(IKK-1)*m_designs_per+m_designs_per,1]))

```

```

a_Resstats[IKK,25] = (a_design_stats[(IKK-1)*m_designs_per+1,1]-
a_design_stats[v_partition_local_min,1])*(a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]-a_design_stats[v_partition_local_min,1])

```

```

a_Resstats[IKK,25] = a_Resstats[IKK,25]-(a_design_stats[(IKK-
1)*m_designs_per+1,1]-Second)*(a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]-Second)

```

```

a_Resstats[IKK,25] =
a_Resstats[IKK,25]/((a_design_stats[v_support,1]-a_design_stats[(IKK-
1)*m_designs_per+1,1])*(a_design_stats[v_support,1]-a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]))

```

```

a_Resstats[IKK,26] = (a_design_stats[(IKK-1)*m_designs_per+1,1]-
a_design_stats[v_partition_local_min,1])*(a_design_stats[v_support,1]-
a_design_stats[v_partition_local_min,1])

```

```

a_Resstats[IKK,26] = a_Resstats[IKK,26]-(a_design_stats[(IKK-
1)*m_designs_per+1,1]-Second)*(a_design_stats[v_support,1]-Second)

```

```

a_Resstats[IKK,26] = a_Resstats[IKK,26]/((a_design_stats[(IKK-
1)*m_designs_per+m_designs_per,1]-a_design_stats[(IKK-
1)*m_designs_per+1,1])*(a_design_stats[(IKK-1)*m_designs_per+m_designs_per,1]-
a_design_stats[v_support,1]))

```

```

a_Resstats[IKK,24] = sqrt(a_Resstats[IKK,24]**2)

```

```

a_Resstats[IKK,25] = sqrt(a_Resstats[IKK,25]**2)

```

```

a_Resstats[IKK,26] = sqrt(a_Resstats[IKK,26]**2)

```

```

# Uses the ratios to determine the allocations

```

```

If ((a_Resstats[IKK,26]<a_Resstats[IKK,24]) and
(a_Resstats[IKK,26]<a_Resstats[IKK,25])) Then

```

```

Begin

```

```

a_Resstats[IKK,27] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,25]/a_Resstats[IKK,24] + a_Resstats[IKK,26]/a_Resstats[IKK,24]))

```

```

If a_Resstats[IKK,27] < 0 Then

```

```

Begin

```

```

a_Resstats[IKK,27] = 0

```

```

End

```

```

If a_Resstats[IKK,27] > a_Resstats[IKK,1] Then

```

```

Begin
  a_Resstats[IKK,27] = a_Resstats[IKK,1]
End

a_Resstats[IKK,28] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,24]/a_Resstats[IKK,25] + a_Resstats[IKK,26]/a_Resstats[IKK,25]))

If a_Resstats[IKK,28] < 0 Then
  Begin
    a_Resstats[IKK,28] = 0
  End

Then
  If (a_Resstats[IKK,27]+a_Resstats[IKK,28]) > a_Resstats[IKK,1]
    Begin
      a_Resstats[IKK,28] = a_Resstats[IKK,1] - a_Resstats[IKK,27]
    End

  If (a_Resstats[IKK,27]+a_Resstats[IKK,28]) < a_Resstats[IKK,1]
    Begin
      a_Resstats[IKK,29] = a_Resstats[IKK,1] - a_Resstats[IKK,27] -
a_Resstats[IKK,28]
    End
  ELSE
    Begin
      a_Resstats[IKK,29] = 0
    End

End

If ((a_Resstats[IKK,24]<a_Resstats[IKK,25]) and
(a_Resstats[IKK,24]<a_Resstats[IKK,26])) Then
  Begin
    a_Resstats[IKK,29] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,25]/a_Resstats[IKK,26] + a_Resstats[IKK,24]/a_Resstats[IKK,26]))

    If a_Resstats[IKK,29] < 0 Then
      Begin
        a_Resstats[IKK,29] = 0
      End

    If a_Resstats[IKK,29] > a_Resstats[IKK,1] Then

```

```

Begin
  a_Resstats[IKK,29] = a_Resstats[IKK,1]
End

a_Resstats[IKK,28] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,24]/a_Resstats[IKK,25] + a_Resstats[IKK,26]/a_Resstats[IKK,25]))

If a_Resstats[IKK,28] < 0 Then
  Begin
    a_Resstats[IKK,28] = 0
  End

  If (a_Resstats[IKK,29]+a_Resstats[IKK,28]) > a_Resstats[IKK,1]
Then
    Begin
      a_Resstats[IKK,28] = a_Resstats[IKK,1] - a_Resstats[IKK,29]
    End

    If (a_Resstats[IKK,29]+a_Resstats[IKK,28]) < a_Resstats[IKK,1]
Then
      Begin
        a_Resstats[IKK,27] = a_Resstats[IKK,1] - a_Resstats[IKK,29] -
a_Resstats[IKK,28]
      End
    ELSE
      Begin
        a_Resstats[IKK,27] = 0
      End
    End

  End

  If ((a_Resstats[IKK,25]<a_Resstats[IKK,24]) and
(a_Resstats[IKK,25]<a_Resstats[IKK,26])) Then
    Begin
      a_Resstats[IKK,27] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,25]/a_Resstats[IKK,24] + a_Resstats[IKK,26]/a_Resstats[IKK,24]))

      If a_Resstats[IKK,27] < 0 Then
        Begin
          a_Resstats[IKK,27] = 0
        End
      End
    End
  End

```

```

    If a_Resstats[IKK,27] > a_Resstats[IKK,1] Then
        Begin
            a_Resstats[IKK,27] = a_Resstats[IKK,1]
        End

        a_Resstats[IKK,29] = Round ((a_Resstats[IKK,1]) / (1+
a_Resstats[IKK,24]/a_Resstats[IKK,26] + a_Resstats[IKK,25]/a_Resstats[IKK,26]))

        If a_Resstats[IKK,29] < 0 Then
            Begin
                a_Resstats[IKK,29] = 0
            End

            If (a_Resstats[IKK,27]+a_Resstats[IKK,29]) > a_Resstats[IKK,1]
Then
                Begin
                    a_Resstats[IKK,29] = a_Resstats[IKK,1] - a_Resstats[IKK,27]
                End

                If (a_Resstats[IKK,27]+a_Resstats[IKK,29]) < a_Resstats[IKK,1]
Then
                    Begin
                        a_Resstats[IKK,28] = a_Resstats[IKK,1] - a_Resstats[IKK,27] -
a_Resstats[IKK,29]
                    End
                    ELSE
                        Begin
                            a_Resstats[IKK,28] = 0
                        End
                    End

                End

            m_OCBA          INT TEMPOCBA

# Initialized to zero to make sure all are assigned
a_Resstats[v_best_partition,1] = 0

        TEMPOCBA = 1
        WHILE TEMPOCBA < (m_partitions+1) DO
            BEGIN
                a_Resstats[TEMPOCBA,1] = 0
                INC TEMPOCBA
            END

```

```

# This block determines the coefficients for each design using OCBA
v_CoefSum = 0
v_CoefSqr = 0

TEMPOCBA = 1
WHILE TEMPOCBA < (m_partitions+1) DO
  BEGIN
    IF TEMPOCBA <> v_best_partition Then
      Begin
        a_Resstats[TEMPOCBA,14] =
(a_Resstats[v_Second_Lowest,13]/a_Resstats[TEMPOCBA,13])**2
# uses ratio rule for nonbest partitions
        v_CoefSum = v_CoefSum + a_Resstats[TEMPOCBA,14]
# checks to see if it is a "full" or "local" comparison, if so, adds to the square root rule for
OCBA
        If a_Resstats[TEMPOCBA,12] > 1 Then
          Begin
            v_CoefSqr = v_CoefSqr +
(a_Resstats[TEMPOCBA,14]/a_Resstats[TEMPOCBA,22])**2
          End
        End
      Else
        Begin
          a_Resstats[TEMPOCBA,14] = 0
        End
      INC TEMPOCBA
    END

# Checks to see if none of the other partitions are using "full" or "local". If so, then just
uses ratio rule for all.
    If v_CoefSqr = 0 Then
      Begin
        a_Resstats[v_best_partition,14] =
(a_Resstats[v_Second_Lowest,13]/a_Resstats[v_best_partition,8])**2
        v_CoefSum = v_CoefSum + a_Resstats[v_best_partition,14]
      End

# This block makes the allocations according to the OCBA coefficients
    Int TempAvail
    Int TempNew
    Int TempSL
    TempAvail = m_delta*m_partitions

```



```

TempNew =
Round(Real(m_delta*m_partitions)/(a_Resstats[v_best_partition,22]*sqrt(v_CoefSqr)+v
_CoefSum))

```

```

If TempNew < 1 Then
Begin
TempNew = 1
End

```

```

If TempNew > TempAvail Then
Begin
TempNew = TempAvail
End

```

```

TempAvail = TempAvail - TempNew
TempSL = TempNew
a_Resstats[v_Second_Lowest,1] = TempNew

```

```

TEMPOCBA = 1
WHILE TEMPOCBA < (m_partitions+1) DO
BEGIN
IF TEMPOCBA <> v_Second_Lowest Then
Begin
IF TempAvail > 0 Then
Begin
TempNew =
Round(a_Resstats[TEMPOCBA,14]*Real(TempSL))
If TempNew > TempAvail Then
Begin
TempNew = TempAvail
End
TempAvail = TempAvail - TempNew
End
Else
Begin
TempNew = 0
End
a_Resstats[TEMPOCBA,1] = TempNew
End
INC TEMPOCBA
END

```

```

If TempAvail > 0 Then
Begin

```

```

                                a_Resstats[v_best_partition,1] = a_Resstats[v_best_partition,1] +
TempAvail
                                End
m_d_opt_runs      10
m_minx            3
m_maxx            8
m_designs         60
m_partitions      6
m_designs_per     m_designs/m_partitions
m_increments      1+(2000 - m_designs * m_nzero)/m_delta
NDIM              3

```

```

*****
*                               *
*               External Files   *
*                               *
*****

```

ID	Type	File Name	Prompt
(null)		a_Results.xls	
(null)		a_scatter.xls	
(null)		f1 function values.xlsx	

## REFERENCES

## REFERENCES

- Atkinson, A.C., A. N. Donev. 1998. *Optimum Experimental Designs*. Oxford Science Publications, Oxford.
- Banks, J., J.S. Carson, B.L. Nelson, D.M. Nicol. 2001. *Discrete –Event System Simulation*, Prentice Hall, Upper Saddle River, New Jersey.
- Barton, R. R. 2005. Issues in Development of Simultaneous Forward-Inverse Metamodels. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 209-217.
- Bazarra, M.S., H.D. Sherali, C.M. Shetty. 1993. *Nonlinear Programming, Theory and Algorithms, 2<sup>nd</sup> Edition*, John Wiley & Sons, New York, New York.
- Bechhofer, R.E., T. J. Santner, D. M. Goldsman. 1995. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. John Wiley & Sons, New York.
- Branke, J., S. Chick, C. Schmidt. 2005. New Developments in Ranking and Selection: An Empirical Comparison of Three Main Approaches. Presentation Slides from the 2005 Winter Simulation Conference.
- Branke, J., S. Chick, C. Schmidt. 2007. Selecting a Selection Procedure. *Management Science* **53**(12) 1916-1932.
- Brantley, M. W., J. McFadden, M. J. Davis. 2002. Expanding the Trade Space: An Analysis of Requirements Tradeoffs Affecting System Design, *Acquisition Review Quarterly* **9**(1) 1-16.
- Brantley, M. W. 2005. The Special Case OCBA Method, working paper, Department of Systems Engineering and Operations Research, George Mason University.
- Brantley, M. W., C. H. Chen. 2005. A Moving Mesh Approach for Simulation Budget Allocation on Continuous Domains. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 699-707.

- Brantley, M. W. 2007. Simulation Budget Allocation for Stochastic Problems on Continuous Domains: Integrating Optimal Computing, Response Surfaces, and Domain Partitioning, working paper, Department of Systems Engineering and Operations Research, George Mason University.
- Brantley, M. W., L. H. Lee, C. H. Chen, A. Chen. 2008. Optimal Sampling in Design of Experiment for Simulation-based Stochastic Optimization, *Proceedings of 2008 IEEE Conference on Automation Science and Engineering*, Washington, DC, 388-393.
- Brantley, M. W. L. H. Lee, C. H. Chen, A. Chen. 2010. Efficient Simulation Budget Allocation with Regression, submitted to *IIE Transactions*.
- Brantley, M. W. L. H. Lee, C. H. Chen. 2011. An Efficient Simulation Budget Allocation Method Incorporating Regression for Partitioned Domains, preparing for submission to *Operations Research*.
- Burden, F., J. Faires. 1993. *Numerical Analysis, Fifth Edition*. PWS Publishers, Boston, Massachusetts.
- Calvin, J.M., A Žilinskas. 2005. One-Dimensional Global Optimization for Observations with Noise. *Computers & Mathematics with Applications*, **50**(1-2) 157-169.
- Chang, K., L. Hong, H. Wan. 2007. Stochastic Trust Region Gradient-free Method (STRONG)- A New Response-Surface-Based Algorithm in Simulation Optimization. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds. *Proceedings of the 2007 Winter Simulation Conference*. IEEE, Piscataway, NJ 346-354.
- Chen, A., J. Cheng. 2006a. Experimental Design for Process Improvement: R-optimum Sequential Design. Unpublished dissertation, National Taiwan University, Taipei, Taiwan.
- Chen, A., J. Cheng. 2006b. An Efficient Estimate of Response Surface Confidence Interval and Its Applications to Sequential Semiconductor Yield Improvement. Working paper, National Taiwan University, Taipei, Taiwan.
- Chen, C. H., J. Lin, E. Yücesan, S. E. Chick. 2000. Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization. *Journal of Discrete Event Dynamic Systems: Theory and Applications* **10** 251-270.
- Chen, C. H., D. He, M. Fu, L. H. Lee. 2008. Efficient Simulation Budget Allocation for Selecting an Optimal Subset. *Inform Journal on Computing* **20**(4) 579-595.

- Chen, C. H., E. Yücesan, L. Dai, H. C. Chen. 2010. Efficient Computation of Optimal Budget Allocation for Discrete Event Simulation Experiment. *IIE Transactions* **42**(1) 60-70.
- Chen, C. H., L. H. Lee. 2010. *Stochastic Simulation Optimization, An Optimal Computing Budget Allocation*. World Scientific Co. Inc, Hackensack, New Jersey.
- Cheng, R. C. H., J. P. C. Kleijnen. 1999. Improved design of queueing simulation experiments with highly heteroscedastic responses. *Operations Research* **47**(5) 762-777.
- Cheng, R. C. H, V. B. Melas, A. N. Pepelyshev. 2001. Optimal Designs for the Evaluation of an Extreme Point. A. Atkinson, B. Bogacka, A. Zhigljavsky, eds. *Optimum Design 2000*. Kluwer Academic Publishers, The Netherlands 1-12.
- Chick, S., K. Inoue. 2001a. New Two-Stage and Sequential Procedures for Selecting the Best Simulated System. *Operations Research* **49** 1609–1624.
- Chick, S., K. Inoue. 2001b. New Procedures to Select the Best Simulated System Using Common Random Numbers. *Management Science* **47**(8) 1133-1149.
- Clemen, Robert T. 1996. *Making Hard Decisions: An Introduction to Decision Analysis, 2<sup>nd</sup> Edition*. Duxbury Press, Pacific Grove, California.
- DeGroot, M. H. 1970. *Optimal Statistical Decisions*. McGraw Hill, New York.
- Dembo, A., O. Zeitouni. 1998. *Large Deviations Techniques and Applications*. Springer, New York.
- Detle, H., V. B. Melas. 2010. A Note on the de la Garza Phenomenon for Locally Optimal Designs. Sonderforschungsbereich (SFB) 823 Discussion Paper.
- De la Garza, A. 1954. Spacing of Information in Polynomial Regression. *The Annals of Mathematical Statistics* **25**(1) 123-130.
- Draper, N. R., H. Smith. 1998. *Applied Regression Analysis: 3rd Edition*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York.
- Dudewicz, E. J., S. R. Dalal, 1975. Allocation of Observations in Ranking and Selection with Unequal Variances. *Sankhya* **B37** 28-78.
- Ehrenfeld, S. 1955. Complete Class Theorems in Experimental Design. *Proceedings Third Berkley Symposium* **1** University of California Press 57-67.

- Fedorov, V. V., W. C. Muller. 1997. A reparametrization view of optimal design for the extremal point in polynomial regression. *Metrika* **46** 147-157.
- Frazier, P., W. B. Powell, S. Dayanik. 2008. A Knowledge-gradient policy for Sequential Information Collection, *SIAM Journal on Control and Optimization* **47**(5) 2410-2439.
- Frazier, P., W. B. Powell, S. Dayanik. 2009. The Knowledge Gradient Policy for Correlated Normal Beliefs, *INFORMS Journal on Computing* **21**(4) 599-613.
- Friedman, J. H. 1991. Multivariate Adaptive Regression Splines, *The Annals of Statistics* **19**(1) 1-67.
- Fu, M.C., F. Glover, J. April. 2005. Simulation Optimization: A Review, New Developments, and Applications. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 584-590.
- Gill, J. 2002. *Bayesian Methods: A Social and Behavioral Sciences Approach*. Chapman & Hall, Boca Raton, Florida.
- Giordana, F. R., M.D. Weir. 1985. *A First Course in Mathematical Modeling*. Brooks/Cole Publishing Company, Monterey, California.
- Glynn, P.W., S. Juneja. 2004. *A Large Deviations Perspective on Ordinal Optimization*. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds. *Proceedings of the 2004 Winter Simulation Conference*. IEEE, Piscataway, NJ 577-585.
- Goldsman, D., B. L. Nelson. 1994. Ranking, Selection, and Multiple Comparison in Computer Simulation. J. Tew, M. S. Manivannan, D. A. Sadowski, A. F. Seila, eds. *Proceedings of the 1994 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 192-199.
- Goldsman, D., S.-H. Kim, B. L. Nelson. 2005. Statistical Selection of the Best System. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 178-187.
- Harrell, C.R., R.E. Bateman, T.J. Gogg, J.R.A. Mott. 1995. *System Improvement Using Simulation, 3<sup>rd</sup> Edition*. ProModel Corporation, Orem, Utah.
- He, D., S. E. Chick, C. H. Chen. 2007. The Opportunity Cost and OCBA Selection Procedures in Ordinal Optimization. *IEEE Transactions on Systems, Man, and Cybernetics--Part C* **37**(5) 951-961.

- Henderson, S. G., A.J. Mason. 2005. Ambulance Service Planning: Simulation and Data Visualization, *International Series in Operations Research & Management Science* **70**(2) 77-102.
- Hsieh, B.W., C. H. Chen, S. C. Chang. 2001. Scheduling Semiconductor Wafer Fabrication by Using Ordinal Optimization-Based Simulation, *IEEE Trans. Robotics and Automation* **17**(5) 599-608.
- Hsieh, B. W., C. H. Chen, S. C. Chang. 2007. Efficient Simulation-based Composition of Dispatching Policies by Integrating Ordinal Optimization with Design of Experiment, *IEEE Trans. Automation Science and Engineering* **4**(4) 553-568.
- Huang, D., T. T. Allen, W. I. Notz, R. A. Miller. 2006. Sequential Kriging Optimization Using Multiple-fidelity Evaluations. *Structural and Multidisciplinary Optimization* **32**(5) 369-382.
- Kiefer, J. 1959. Optimum Experimental Designs. *Journal of the Royal Statistical Society, Series B (Methodological)* **21**(2) 272-319.
- Kim, S.-H., B. L. Nelson. 2006. Selecting the best system. S.G. Henderson, B.L. Nelson, eds. Chapter 18 in *Handbooks in Operations Research and Management Science: Simulation*. Elsevier, Amsterdam, The Netherlands.
- Kushner, H. J. 1964. A New Method of Locating the Maximum of an Arbitrary Multi-peak Curve in the Presence of Noise. *Journal of Basic Engineering* **86** 97-106.
- Lamb, J. D., R. C. H. Cheng. 2002. Optimal allocation of runs in a simulation metamodel with several independent variables. *Operations Research Letters* **30** 189-194.
- Law, A. M., W. D. Kelton. 2000. *Simulation Modeling and Analysis*, McGraw-Hill, Inc.
- Liski, E. P., N. Mandal, K. Shah, B. Sinha. 2001. *Lecture Notes in Statistics: Topics in Optimal Design*, Springer, New York.
- Melas, V. B., A. Pepelyshev, R. C. H. Cheng. 2003. Designs for estimating an extremal point of quadratic regression models in a hyperball. *Metrika* **58** 193-208.
- Melas, V. B. 2006. *Functional Approach to Optimal Experimental Design. Lecture Notes in Statistics 184*. Springer, New York.
- Morrice, D. J., M. W. Brantley, C. H. Chen. 2008. An Efficient Ranking and Selection Procedure for a Linear Transient Mean Performance Measure. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, eds. *Proceedings of the 2008 Winter Simulation Conference*. IEEE, Piscataway, NJ 290-296.



- Morrice, D. J., M. W. Brantley, C. H. Chen. 2009. A Transient Means Ranking and Selection Procedure with Sequential Sampling Constraints. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds. *Proceedings of the 2009 Winter Simulation Conference*. IEEE, Piscataway, NJ 590-600.
- Neddermeijer, H., G.J. van Oortmarssen, N. Piersma. 2000. A Framework for Response Surface Methodology for Simulation Optimization. J.A. Joines, R.R. Barton, K. Kang, P.A. Fishwick, eds. *Proceedings of the 2000 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 129-136.
- Neter, J., M. Kutner, C. Nachtsheim, W. Wasserman. 1996. *Applied Linear Regression Models, 3rd Edition*. Times Mirror Higher Education Group.
- Rinott, Y. 1978. On Two-stage Selection Procedures and Related Probability Inequalities. *Communications in Statistics* **A7** 799-811.
- Sanchez, S. M. 2005. Work Smarter, Not Harder: Guidelines for Designing Simulation Experiments. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*. IEEE, Piscataway, New Jersey 69-82.
- Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, John Wiley and Sons, Hoboken, New Jersey, 215.
- Törn, A., A. Žilinskas. 1989. Global optimization. G. Goos, J. Hartmanis, eds. *Lecture Notes in Computer Science* **350** Springer-Verlag, Berlin.
- van Beers, W. C. M., J. P. C. Kleijnen. 2003. Kriging for Interpolation in Random Simulation. *Journal of the Operational Research Society* **54** 255-262.
- Villemonteix, J., E. Vazquez, E. Walter. 2009. An Informational Approach to the Global Optimization of Expensive-to evaluate Functions. *Journal of Global Optimization* **44**(4) 509-534.
- Yang, M. 2010. On the de la Garza Phenomenon. *Annals of Statistics*. Institute of Mathematical Sciences, **38**(4) 2499-2524.
- Zeto, J., M. W. Brantley, G. Collins, et al. 2005. "Air Ambulance Analysis-Iraq", presented at the 2005 Military Operations Research Symposium, West Point, New York.

## CURRICULUM VITAE

Mark W. Brantley received a BS in Mathematics from the United States Military Academy in 1988, an MS in Applied Mathematics from Rensselaer Polytechnic Institute in 1998, and an MS in Operations Research and Statistics also from Rensselaer Polytechnic Institute in 1998.

Mark retired from the US Army after completing a 20 year career which included deployments to Kuwait in support of Operation Enduring Freedom and Operation Iraqi Freedom; to Haiti in support of Operation Uphold Democracy; and to Honduras for service with Joint Task Force - Bravo. Mark's career also included service as an Operations Research analyst in the Operations Research Center of Excellence at the United States Military Academy, at the Center for Army Analysis, and in the Army National Guard Directorate.

Mark is the co-recipient of the 2005 and 2006 Military Operations Research Society Rist Prizes for "Air Ambulance Analysis-Iraq" and "Army Force Generation Model Simulation" respectively. These prizes are awarded to the best implemented DoD military operations research study from those submitted in response to a call for papers. He is also the co-recipient of the 2004 Army Operations Research Symposium Payne Award also for "Air Ambulance Analysis-Iraq."

After retiring from the US Army, Mark worked for two years as a senior defense forecast analyst for Documental Solutions and Jane's Information Group. He has formed his own company called Goose Point Analysis LLC specializing in modeling and simulation, data analysis, and forecasting and will begin to build a client base in the summer of 2011.

In the course of study at George Mason University, Mark contributed to three sections in Chen and Lee (2010), co-authored two journal papers that have been submitted (Brantley et al 2010 and Brantley et al. 2011), and co-authored four conference papers (Brantley and Chen 2005, Brantley, Lee, Chen, and Chen 2008, Morrice, Brantley, and Chen 2008, Morrice, Brantley, and Chen 2009). In addition to these publications, Mark is the co-author of one refereed journal article, eight US Army technical reports, and four magazine articles. He has provided presentations at the Army Simulation and Modeling for Acquisition, Requirements, and Training (SMART) conference, the Army Operations Research Symposium (AORS), the INFORMS annual meeting, and the Winter Simulation Conference.